Лекция 10

Вероятностный алгоритм для 3-SAT. Приближенный алгоритм для минимального вершинного покрытия. Вычисление максимального потока.

(Конспект: М. Плискин)

10.1 Решение задачи 3-SAT

В данном разделе будет приведен вероятностный алгоритм для решения задачи 3-SAT (3-выполнимость).

Определение 10.1 (3-SAT). Пусть есть логическая формула, записанная как конъюнкция элементарных дизъюнкций, каждая из которых содержит в себе не более трех литералов (переменных либо их отрицаний). Необходимо найти выполняющий набор значений переменных, то есть такую подстановку, на которой наша формула истинна.

Будем считать, что значения всех наших переменные хранятся в «массиве», значение i-ой переменной — $A[i]$. Опишем наш вероятностный алгоритм.

Алгоритм 10.1. Инициализируем массив A случайными значениями.

Если A — выполняющий набор, то завершаем работу. В противном случае существует c — ложная дизъюнкция при наборе A:

$$C = x_1 \vee x_2 \vee x_3,$$

где x_1, x_2, x_3 — переменные либо их отрицания. Изменим значение в одной из переменных (случайным образом выбранной) на противоположное. Проделаем это $3n$ раз (внутренний цикл).

Всю описанную процедуру (начиная с выбора случайного A) снова повторим некоторое экспоненциальное количество раз (точное число определим позднее). □

Теперь зададимся вопросом, сколько же раз нам необходимо повторить вышеописанную процедуру. Для этого рассмотрим элементарный шаг и отметим, что вероятность уменьшить расстояние $d(A, S)$ от нашего набора A до выполняющего набора S, понимаемое как количество позиций, в которых биты не совпадают, не менее 1/3. Это следует из того, что на выполняющем наборе рассматриваемая дизъюнкция C должна быть истинной. Поэтому хотя бы один из литералов должен быть истинным. Таким образом, с вероятностью по крайней мере 1/3 мы выберем правильный литерал. Таким образом, процесс выполнения алгоритма можно рассматривать как случайное блуждание на отрезке $[0..n]$, где n — число переменных в формуле, с отражением на правом конце и поглощении при попадании в начало.

Определим величину $P(t, j)$ как вероятность попасть из состояния j (расстояние j) в состояние 0 (т.е. конечное состояние) за t шагов. Далее представим t в виде $t = j + 2i$. (Очевидно, i — целое число.)

Упражнение 10.1. Доказать, что

$$P(t, j) = \binom{t}{i} \frac{j}{t} \left(\frac{1}{3}\right)^{j+i} \left(\frac{2}{3}\right)^{i}.$$

Указание. Первый вариант. Написать рекуррентное уравнение для путей на плоскости из точки $(0, j)$ в точку $(t, 0)$, лежащих полностью в первом квадранте (и не касающихся осей за исключением концов); на каждом шаге пути мы сдвигаемся на вектор $(1, -1)$ либо $(1, 1)$. Далее доказать по индукции, что этих путей $\binom{t}{i}\frac{j}{t}$.
Второй вариант. Взять книжку Феллера и посмотреть более простое док-во (см. Теорему о баллотировке).

Теперь определим величину $P(j)$ как вероятность попасть в 0 за какое-нибудь количество шагов:

$$P(j) = \sum_t P(t, j).$$

Оценим эту сумму снизу ее слагаемым, соответствующим $j = i$:

$$P(j) \geq \frac{1}{3}\binom{3j}{j}\left(\frac{1}{3}\right)^{2j}\left(\frac{2}{3}\right)^{j} \geq$$
$$\geq \frac{1}{p(n)} \cdot \frac{(3j)^{3j}}{j^j(2j)^{2j}} \cdot \left(\frac{1}{3}\right)^{2j} \cdot \left(\frac{2}{3}\right)^{j} =$$
$$\frac{1}{p(n)}\left(\frac{3^3 \cdot j^3}{j \cdot 2 \cdot 2^2 \cdot j^2} \cdot \frac{1}{3^2} \cdot \frac{2}{3}\right)^{j} == \frac{1}{p(n)} \cdot \frac{1}{2^j},$$

где $p(n)$ — некоторый многочлен степени n.

Отметим, что в нашей оценке

$$t = j + 2i = 3j \leq 3n,$$

так как $j \leq n$; т.е., она верна, если внутренний цикл повторять $3n$ раз. Теперь вычислим вероятность успеха, то есть вероятность попасть в 0.

$$P_0 = \sum_{j=0}^{n} Q(j) \cdot P(j),$$

где

$$Q(j) = \frac{1}{2^n}\binom{n}{j}$$

— вероятность оказаться в состоянии j при начальном выборе массива A. Подставим:

$$P_0 = \frac{1}{2^n p(n)} \sum_{j=0}^{n} \binom{n}{j} \frac{1}{2^j} = \frac{1}{p(n)}\left(\frac{1}{2}\left(1 + \frac{1}{2}\right)\right)^{n} = \left(\frac{3}{4}\right)^{n} \cdot \frac{1}{p(n)}.$$

Как известно, в этом случае для получения константной вероятности ошибки надо взять количество итераций не менее

$$\frac{1}{P_0} = p(n)\left(\frac{4}{3}\right)^{n}.$$

10.2 Приближенный алгоритм для минимального вершинного покрытия

Определение 10.2. Пусть $G = (V, E)$ — граф. Назовем множество $A \subseteq V$ *вершинным покрытием графа* G, если для каждого ребра этого графа хотя бы один из его концов принадлежит покрытию.

Рассмотрим взвешенный граф (т.е. каждой вершине сопоставлен некоторый вес). Наша задача состоит в том, чтобы приближенно найти минимальное в смысле веса вершинное покрытие. *Будем постепенно сводить эту задачу к другим, пока не сведем ее к нахождению максимального потока.*

Сначала построим двудольный граф B следующим образом: его множество вершин есть удвоенное множество вершин исходного графа G (т.е. для каждой вершины $v \in V_G$ существуют вершины $v_1, v_2 \in V_B$), а ребра подчиняются следующем правилу: если между исходными вершинами u и v в графе G было ребро, то в графе B ребра будут между u_1 и v_2 и между u_2 и v_1. Ясно, что граф B является двудольным, причем все вершины с индексом «1» принадлежат одной доле, а с индексом «2» — другой.

Определим теперь понятие двойного покрытия графа G.

Определение 10.3. Отображение $f\colon V_G \to \{0, 1, 2\}$ называется *двойным покрытием графа G*, если для каждого ребра либо один из его концов имеет пометку 2, либо оба конца имеют пометку 1.

Лемма 10.1. *Минимальное вершинное покрытие графа B соответствует минимальному двойному покрытию графа G.*

Упражнение 10.2. Доказать лемму 10.1.

Лемма 10.2. *Пусть C — минимальное двойное покрытие G. Тогда его мощность как множества не превосходит удвоенной мощности минимального покрытия G.*

Упражнение 10.3. Доказать лемму 10.2.

Определение 10.4. Пусть G — некоторый связный граф. Назовем *$s-t$-сечением* графа G разбиение его вершин на два множества V_s и V_t, в одном из которых содержится вершина s, а в другом — t. *Вес сечения* — это суммарный вес всех ребер, соединяющих вершины V_s с вершинами V_t.

Добавим в граф B две вершины s и t и соединим вершину s со всеми вершинами одной доли, а t — со всеми вершинами другой. Припишем вновь добавленным ребрам такие же веса, какие были у соответствующих вершин в исходном графе, а старым ребрам — вес $+\infty$.

Лемма 10.3. *Минимальному покрытию графа B соответствует $s-t$-сечение минимального веса для только что построенного графа B'.*

Доказательство. Сопоставим каждому сечению множество вершин — концов его ребер, отличные от s и t. Получим покрытие. Аналогично обратный переход. Веса согласованы по построению. □

Лемма 10.4. *Минимальное $s - t$-сечение находится за время $O(n^3)$.*

Доказательство. Доказательство данной леммы непосредственно следует из следующего утверждения и материала раздела 10.3. □

Сначала дадим определение.

Определение 10.5 (Поток в графе). Пусть есть ориентированный граф G и в нем каждому ребру сопоставлен вес w. Этот вес понимается как пропускная способность данного ребра. Тогда *потоком* из вершины s в вершину t называется пометка ребер графа, удовлетворяющая условию: на каждом ребре значение потока не превосходит его пропускной способности, и, кроме того, сумма потоков на ребрах, входящих в вершину, не превышает суммы потоков на ребрах, выходящих из нее (кроме вершин s и t). Значением потока является сумма (со знаком) пометок ребер с началом (концом) в s.

Теперь сформулируем наше утверждение.

Лемма 10.5. *Максимальный поток определяет минимальное s—t-сечение.*

Доказательство. Действительно, рассмотрим максимальный поток. Вычтем его из пропускной способности ребер. Тогда на каждом пути из s в t найдется (иначе поток по этому пути можно увеличить) нулевое ребро. Его и добавим в сечение. Оно, по построению, минимально. □

10.3 Поиск максимального потока

Пусть у нас есть взвешенный по ребрам граф G, и мы хотим найти максимальный поток из s в t. Сопоставим каждой вершине графа функцию $h(v)$, понимаемую как высоту данной вершины. Сначала мы построим *предпоток* — поток, не удовлетворяющий второму условию определения, то есть в вершине может быть неотрицательный дефект $\Delta(v)$ — превышение входа над выходом.

Мы построим сначала некоторый предпоток, а потом займемся преобразованием его в поток.

Сначала припишем вершине s высоту n, а всем остальным вершинам — высоту 0. Направим поток согласно пропускным способностям каждого из ребер, исходящих из s. Далее займемся «лечением» дефектов по следующему алгоритму:

Алгоритм 10.2. Для каждой вершины v из списка «дефектных» вершин до предела (или пока не пропадет дефект) увеличиваем поток по исходящим ребрам. Делаем это только для исходящих ребер, идущих вниз (по высоте). Если такой возможности больше нет, поднимаем вершину на высоту, минимально возможную для того, чтобы возможность вновь появилась, итд. При этом мы обрабатываем «дефектные» вершины по списку, после каждого подъема отправляя вершину в начало списка и продолжая начиная обход списка с нее (т.е. вновь с начала). □

Лемма 10.6. *Если поток из вершины u в вершину v можно увеличить, то $h(u) \le h(v) + 1$.*

Доказательство. Отметим, что в начале работы алгоритма это свойство выполнено. Далее, при каждом шаге в случае, когда мы не изменяем высоты, соотношение продолжает выполняться. Высоты мы начинаем изменять в том случае, когда без этих изменений исходящий поток увеличить нельзя, т.е. условие леммы не выполняется. Высоты мы изменяем ровно до того момента, чтобы соотношение начало выполняться, и лишь в этом случае появляется возможность дальнейшего увеличения потока. Таким образом, соотношение сохраняется при работе нашего алгоритма. □

Лемма 10.7. *Если мы нашли поток, то он максимален.*

Доказательство. Отметим, что вершина s никогда не опускается, а t никогда не поднимается. У нас остается $n-2$ вершины, а длина пути по высоте составляет $n-1$. Если поток не максимален, то должен существовать путь, по которому можно пропустить еще немного воды, а для него по лемме 10.6 между любыми двумя вершинами разность высот не более 1, то есть суммарный перепад высот не более $n-2$, а должен быть $n-1$. То есть такого пути не существует, а значит поток максимален. □

Заметим, что когда наш алгоритм останавливается, все дефекты равны нулю, т.е. поток (а значит, максимальный поток) мы находим всегда; остается понять, через какое количество шагов наш алгоритм останавливается.

Упражнение 10.4. Доказать, что максимальная высота $h \le 2n$.

Тогда количество подъемов не более $O(n^2)$. Между подъемами не более n шагов (максимальная длина списка «дефектных» вершин). Таким образом, общее «межподъемное» время есть $O(n^3)$. Время, затраченное на сами подъемы, оценивается через их количество (n^2), умноженное на время самого подъема, пропорциональное степени поднимаемой вершины. Можно рассмотреть максимальную степени вершины, и тогда время, затраченное на подъемы, будет не более $O(mn)$, где m — количество ребер, что не превосходит $O(n^3)$. Таким образом, общее время работы алгоритма есть $O(n^3)$.