Лекция 2

Алгоритмы для выполнимости

(Конспект: Д. Ицыксон)

2.1 Постановка задачи

Мы рассматриваем пропозициональные формулы в 3-*КНФ*, то есть конъюнкции дизъюнкций, каждая из которых содержит не более трех литералов (литерал здесь — это переменная или отрицание переменной). Пример: $F = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee z) \wedge (x \vee y)$.

Формула называется *выполнимой*, если существуют такие значения переменных (*выполняющий набор*), что значение формулы будет «истина». Задача выполнимости (в данном случае — 3-SAT) такова: определить, является ли данная формула (в 3-КНФ) выполнимой.

Далее везде n — число переменных формулы.

Есть очевидный алгоритм, решающий эту задачу за время $2^n \cdot \mathrm{poly}(|F|)$ (далее полиномиальный множитель будем опускать), перебирающий все возможные значения переменных.

Эта задача **NP**-полна, поэтому было бы слишком самонадеянно искать полиномиальные алгоритмы для ее решения. Но пытаться получать хорошие (лучше, чем 2^n) экспоненциальные оценки никто не мешает.

Есть три основных подхода. К которым мы и переходим.

2.2 Divide-and-conquer

Идея в этом подходе очень простая: выбираем литерал, допустим l; затем ищем выполняющий набор для формулы $F[l := 1]$ (далее пишем просто $F[l]$). Если не нашли, то для $F[l := 0]$ (далее $F[\neg l]$). После каждой подстановки необходимо сделать упрощение: удалить дизъюнкции, которые стали заведомо истинными, и вычеркнуть заведомо ложные литералы.

Заметим, что если получилась пустая дизъюнкция, то формула невыполнимая, а если формула не содержит дизъюнкций (все вычеркнули), то формула выполнимая.

У нас осталась свобода: выбирать литерал, по которому расщеплять.

Стратегия: выбирать самую короткую дизъюнкцию и расщеплять по одной из ее переменных. Очевидно, что пока дизъюнкция «жива», мы будем расщеплять по ее переменным. Самая короткая дизъюнкция содержит не более трех переменных, то есть, она породит не более 8 вариантов. Но один из вариантов сразу отпадает: который делает эту дизъюнкцию ложной. То есть на время работы мы получаем следующую рекуррентную оценку: $T(n) \leq T(n-3)+\mathrm{poly}(n)$, отсюда $T(n) = O((7^{\frac{1}{3}})^n) = O(1.92^n)$.

Можно получить более точную оценку. Пусть самая короткая дизъюнкция — $x \vee y \vee z$. И расщепления будут следующие: $F[x]$, $F[\neg x, y]$, $F[\neg x, \neg y, z]$, $F[\neg x, \neg y, \neg z]$. Последняя формула заведомо невыполнима. Получаем оценку: $T(n) \leq T(n-1) + T(n-2) + T(n-3) + \mathrm{poly}(n)$. Тут получается $T(n) = O(1.84^n)$.

2.3 Критические дизъюнкции

Определение 2.1. Предположим, что у формулы только один выполняющий набор. Тогда для каждой переменной x в формуле имеется *критическая дизъюнкция*, то есть такая дизъюнкция, что выполняющий набор присваивает 1 только одному литералу в этой дизъюнкции, и именно, литералу, соответствующему переменной x. Переменная x называется *главной* в критической дизъюнкциию. Итого есть n критических дизъюнкций.

Алгоритм 2.1. Берем случайную переменную, присваиваем ей случайное значение. И так далее, но если видим переменную, единственную в своей дизъюнкции, то присваиваем ей то значение, которое выполняет эту дизъюнкцию (а не случайное).

С вероятностью $\frac{1}{2}$ мы угадываем правильное значение переменной. Если нам повезло и в какой-то критической дизъюнкции мы выбирали главную переменную последней, то ей мы присваиваем правильное значение бесплатно. Критическая дизъюнкция «сыграет» с вероятностью $\frac{1}{3}$. То есть математическое ожидание количества «сыгравших» дизъюнкций составляет $n/3$. С вероятностью $\Omega(\frac{1}{n})$ их будет не менее $n/3$ (например, по неравенству Маркова). При этом условии вероятность того, что правильно выбрали значения «угадываемых» переменных составляет $\frac{1}{2^{2n/3}}$

(поскольку лишь $2n/3$ раз угадывали). Вышеуказанные события независимы, итого с вероятностью $\frac{1}{n2^{2n/3}}$ мы найдем выполняющий набор (если он один). Чтобы уменьшить вероятность ошибки до (произвольной) константы, достаточно повторить эту процедуру $2^{2n/3}\mathrm{poly}(n)$ раз.

Теперь считаем, что выполняющих наборов много.

Определение 2.2. Выполняющий набор называется *изолированным с r сторон*, если существует r переменных, таких, что изменение значения любой из них делает его невыполняющим.

Оценим вероятность того, что алгоритм выдаст конкретный выполняющий набор I. Аналогично предыдущим рассуждениям эта вероятность составляет $\frac{1}{2^{n-\frac{r(I)}{3}}}$ с точностью до полиномиального сомножителя. Выполняющих наборов может быть много, поэтому просуммируем эту вероятность по всем наборам:

$$\sum_I \frac{1}{2^{n-\frac{r(I)}{3}}}.$$

Лемма 2.1. $\sum_I 2^{r(I)-n} \geq 1$.

Доказательство. Индукция по числу переменных. База (одна переменная) тривиальна. Переход: возьмем одну переменную x. Разделим все наборы на две группы, в одной $x = 0$, в другой $x = 1$.

Первый случай — когда одна из групп пуста (скажем, вторая), тогда по индукционному предположению имеем:

$$\sum_{I'} 2^{r(I')-n+1} \geq 1,$$

где суммирование ведется по I' — выполняющим наборам формулы $F[x := 0]$. Ясно, что $r(I) = r(I') + 1$, то есть полученное неравенство и есть требуемое.

Второй случай — когда обе группы не пусты. По индукционному предположению для каждой группы:

$$\sum_{I'} 2^{r(I')-n+1} \geq 1 \iff \sum_{I'} 2^{r(I')-n} \geq \frac{1}{2},$$

$$\sum_{I''} 2^{r(I'')-n+1} \geq 1 \iff \sum_{I''} 2^{r(I'')-n} \geq \frac{1}{2}.$$

Суммируя, получаем требуемое.

$\square$

Далее,

$$\sum_I \frac{1}{2^{n-\frac{r(I)}{3}}} = \sum_I 2^{\frac{r(I)-n}{3}-\frac{2n}{3}} = 2^{-\frac{2n}{3}} \sum_I 2^{\frac{r(i)-n}{3}} \geq 2^{-\frac{2n}{3}} \sum_I 2^{r(i)-n} \geq 2^{-\frac{2n}{3}}.$$

Таким образом, получили такую же оценку, как и в случае единственного выполняющего набора.

2.4 Локальный поиск

Алгоритм 2.2. Начинаем с какого-нибудь (случайного) набора значений переменных. Если он невыполняющий, то есть дизъюнкция, в которой все литералы ложные, выберем из них случайно один и изменим его значение. Повторяем T раз.

Нетрудно показать, что для 2-КНФ среднее время работы $O(n^2)$. Вернемся к 3-КНФ. В метрике Хемминга (расстояние между наборами — число несовпадающих значений) с вероятностью $2/3$ мы удаляемся на 1 от выполняющего набора, с вероятностью $1/3$ — приближаемся.

Лемма 2.2. *Если мы находимся на расстоянии i от выполняющего набора, то вероятность за $3i$ шагов в него попасть не менее* $\frac{1}{2^i}$

Упражнение 2.1. Доказать лемму 2.2.

Вероятность того, что мы наткнемся на выполняющий набор за разумное число шагов, можно оценить следующим образом:

$$\sum_i \frac{1}{2^i}\frac{C_n^i}{2^n} \geq \frac{C_n^{n/3}}{2^n 2^{n/3}} = \frac{1}{\mathrm{poly}(n)}\left(\frac{3}{4}\right)^n.$$

Последнее равенство получается непосредственным применением формулы Стирлинга. Следовательно, за $\left(\frac{4}{3}\right)^n \cdot \mathrm{poly}(n)$ шагов алгоритм найдет выполняющий набор с любой наперед заданной константной вероятностью.

Замечание 2.1. В этом алгоритме можно избавиться от случайных чисел. Для этого пространство всех наборов покрывается «небольшими» шариками в метрике Хемминга; начальные наборы — центры этих шариков. Правда, в результате получается время $(3/2)^n$, поскольку случайное блуждание также надо дерандомизировать.