Взаимосвязь теории кодирования и теории сложности вычислений

Инютин Дмитрий

Аннотация

Взаимодействие теории кодирования и теории сложности вычислений это богатый источник интересных результатов и проблем. В этой статье мы покажем, как одно используется в другом и наоборот. Для этого мы рассмотрим следующие две темы :

1. Как задачи теории сложности используются в криптографии и влияют на нашу ежедневную жизнь.
2. Использование различных кодов для получения характеристик традиционных сложностных классов, таких как NP и $PSPACE$. Эти характеристики, в свою очередь, используются, чтобы показать, что некоторые комбинаторные задачи оптимизации так же трудно приблизить, как и решить точно.

1 Вступление

Мы предполагаем, что читатель знаком с основными терминами теории сложности вычислений. Однако, дадим некоторое представление о теории кодирования.

Определение 1.1. Кодом над алфавитом Σ называется функция E из Σ^k в Σ^n, где k, $n \in N$. Если $\Sigma = \{0, 1\}$, то такой код называется **бинарным**. Слова в области определения E называются **сообщениями**, а слова в образе E называеются **кодовыми словами**.

Определение 1.2. Кодированием слова $A \in \Sigma^k$ называется вычисление $E(A)$. **Обнаружить ошибку** значит по данному слову $B \in \Sigma^n$ определить, существует ли слово $A \in \Sigma^k$ такое что $E(A) = B$. Предъявить такое сообщение A значит **декодировать**.

На основе теории кодирования созданы различные алгоритмы. Они используются для преобразования данных из формы, удобной для непосредственного использования, в форму, удобную для передачи, хранения, автоматической переработки и сохранения от несанкционированного доступа.

Рассмотрим пример.

2 Криптография

Криптография — наука о методах обеспечения конфиденциальности (невозможности прочтения информации посторонним), целостности данных (невозможности незаметного изменения информации), аутентификации (проверки подлинности авторства или иных свойств объекта), а также невозможности отказа от авторства.

Существование этой науки мотивироварованно следующей проблемой. Предположим, что Алиса и Боб хотят передавать сообщения друг другу. Причем каждый из них должен иметь возможность удостовериться, что полученное сообщение корректно и действительно отправлено собеседником. Эта проблема

едва ли не с изобретения письменности. Так, Гай Юлий Цезарь использовал знаменитый Шифр Цезаря для секретной переписки со своими генералами. В Средние века криптография начинает широко использоваться дипломатами, купцами и даже простыми гражданами. Постепенно, по мере распространения техники частотного криптоанализа, шифры усложняются. Перед началом Второй мировой войны ведущие мировые державы имели электромеханические шифрующие устройства, результат работы которых считался невскрываемыми. Однако взлом «Энигмы», использовавшийся сухопутными войсками Германии и её союзниками, позволил изменить исход войны. Наконец, современный период развития криптографии отличается зарождением и развитием нового направления — криптография с открытым ключом.

Односторонняя функция - это функция $f : \{0,1\}^* \to \{0,1\}^*$ такая что существует детерминированная машина Тьюринга, на которой $f(x)$ вычисляется за полиномиальное время на любом входе, но не существует полиномиальной вероятностной машины Тьюринга, которая обращала бы эту функцию с более чем экспониционально малой вероятностью. То есть для любой вероятностной полиномиальной машины M, для любого полинома $p(n)$ и достаточно большого $n \in N$ выполняется:

$$P_r[M(f(m)) \in f^{-1}(m)] < \frac{1}{p(n)}$$

где строка m выбирается случайным образом на множестве $\{0,1\}^n$ в соответствии с равномерным законом распределения. Время работы машины M ограничено полиномом от длины искомого прообраза.

Существование односторонних функций не доказано. Если f является односторонней функцией, то нахождение обратной функции является трудновычислимой, но легкопроверяемой задачей. Таким образом из существования односторонней функции следует, что $P \neq NP$. Однако сами такие функции нас не интересуют. С помощью них можно зашифровать сообщение, но расшифровать потом обратно нельзя. Поэтому используют лазейку - некий секрет, который поможет расшифровать сообщение, т.е. y, такой что, зная y и $f(x)$, можно легко (за полиномиальное время) восстановить x. Классическим примером такой функции является дискретное логарифмирование. Пусть G это некоторая группа и $g \in G$. Определим для любого натурального k g^k, как $g \cdot g \cdot g ... \cdot g$ k раз. Тогда можно ввести операцию логарифма. Пусть $y \in G$. Тогда $log_g y = x \leftrightarrow g^x = y$.
Перейдем к постановке задачи. Пусть G и g, $y \in G$ заданы. Решение задачи дискретного логарифмирования состоит в нахождении некоторого целого неотрицательного числа x, удовлетворяющего уравнению $g^x = y$. Оказывается, что для любого x вычислить g^x можно за полиномиальное время. Действительно, воспользуемся тем, что $g^{ab} = g^a g^b$. Тогда $g^x = (g^{\frac{x}{2}})^2 = (g^{\frac{x}{4}})^4$. Таким образом, зная x можно легко проверить, что он является решением уравнения $g^x = y$, т.е. x является сертификатом. Итого, задача дискретного логарифмирования лежит в NP. Однако науке не известны алгоритмы, решающие эту задачу за полиномиальное время.
Осталось описать, как с помощью описанного математического аппарата зашифровать сообщения. Пусть Алиса и Боб выбирают некоторое общее $g \in G$. Этот элемент группы естественно доступен и третьему лицу, который хочет перехватывать сообщения(Ева). Теперь, Алиса и Боб выбирает по приватному ключу h_a, h_b соответственно. Каждый из них генерирует публичный ключ $H_a = g^{h_a}$, $H_b = g^{h_b}$. Они обмениваются ими по незащищенному каналу. Однако таким образом они получают общий секретный ключ: $S = H_b^{h_a} = H_a^{h_b} = g^{h_a h_b}$. При этом Ева имеет только g, H_a, H_b и по ним, как мы показали выше она не сможет получить S, веди иначе она умела бы решать задачу дискретного логарифмирования. Получив общий секретный ключ, Алиса и Боб могут обмениваться данными с симметричным шифрованием.

3 Использование полиномиальных кодов для описания сложностных классов и доказывания нижней границы

Ранее мы показали, что различные сложные задачи лежат в основе криптографии. Можно попробовать решить их с некоторой заданной наперёд точностью. Однако оказывается, что некоторые задачи также трудно приблизить, как и решить.

Определение 3.1. полиномиальным кодом над конечным полем F называется такой код, сообщения которого представлены как коэффициенты полинома $p(x_1, ..., x_m)$ над F. Кодовое слово, соответствующее сообщению A получается записью значений полинома p для каждого набора длины m в F^m или в некотором подмножестве F^m.

Babai, Fortnow и Lund [1] обнаружили новую характеристику для класса $NEXP$, показав, что каждый язык, принимаемый недетерминированной экспоницеальной по времени машиной Тьюринга также принимается системой с "мульти-интерактивным доказательством"(СМИП). В СМИП два или больше вычислимо-неограниченных "доказывающих"(provers) убеждают вероятностный полиномиальный по времени верификатор, что входная строка x лежит в языке L. Если L это произвольный язык из $NEXP$, то принятый путь вычисления x на недетерминированной машине Тьюринга для языка L имеет экспоненциальную от размера входа x длину. Однако контринтуитивным выглядит тот факт, что правильность такого вычисления может быть проверена на полиномиальной по времене машине Тьюринга. В частности, такая машина не может прочитать полностью всё вычисление, то есть вход. Доказывающие преодолевают это, используя кодирование с помощью полиномиальных кодов. Основным свойством правильного кодирования заключается в том, что оно использует полиномы нескольких переменных соответствующей степени над соответствующим полем. Вероятностный полиномиальный по времени верификатор затем использует специальный алгоритм тестирования полиномов нескольких переменных, чтобы проверить, что закодированые слова соответствуют принимающему вычислению. Такой тестирующий алгоритм выполняет выборку полинома в случайных точках в своей области определения и не обязан читать всё кодирование экспониальной длины.

Результаты, обсуждаемые ниже, в значительной степени отражают методы, изложенные в [1], и представляют расширение возможностей использования теории кодирования в новых характеристиках традиционных классов сложности. Мы покажем, как полученные характеристики NP и $PSPACE$ могут быть использованны для доказательства нижних границ сложности аппроксимаций некоторых комбинаторных функций оптимизации. Точнее, существует $\epsilon > 0$, для которого аппроксимация с точностью до ϵ лежит среди NP-полных, либо PSPACE-полных сооответственно, то есть её также трудно вычислить, как и саму функцию. Эта нижняя граница является таковой в том же смысле, в котором результат NP-полноты или $PSPACE$-полноты является нижней границей: пока $P \neq NP$ эти аппроксимации не могут быть вычислены за полиномиальное время.

Недетерминированная полиномиальная по времени машина Тьюринга M может быть рассмотрена как система доказательств в которой полиномиальный верификатор детерминирован, а утверждения имеют вид "$x \in L(M)$". Если x лежит в $L(M)$, то принимаемое вычисление машины M на входе x и есть доказательство, а если x не принадлежит $L(M)$, то такого доказательства не существует. Заметим, что доказательства с помощью такой системы очень хрупкие: если один бит в описании вычисления меняется, то решение принять или отклонить x в этом вычислении также может измениться. Цель таких систем доказательств - сделать доказательства нахождения языка в NP, более сильными: Если правильное доказательство слегка изменено, оно всё равно должно распознаваться как "по существу правильное"и вход x, которого нет в языке должен приводить к вычислениям "далеких от правильных". Это тесно связано с целями обнаружения и исправления ошибок, что адресует нас к традиционной теории кодирования.

Проверка нахождения языка в некотором классе, которую мы здесь определяем называют **веро-**

ятностным проверяемым доказательством и впервые были формально определены Arora-ой и Safra-ой. Язык L находится в $PCP(r(n), q(n))$, если есть вероятностная машина V (называемая верификатором) с произвольным доступом к ленте со случайными битами. Входом к V является пара (x, y). Утверждается, что строчка x принадлежит L и y это предполагаемое доказательство данного утверждения. В процессе вычисления на входе (x, y) V подбрасывает $O(r(n))$ монет и проверяет $O(q(n))$ бит доказательства строки y, где n это длина x. Если x принадлежит L, то существует доказательство - строка y - такое что V выдаёт 1 с вероятностью 1 на входе (x, y). Если $x \neq L$, то для всех строк y, V выдаёт 1 с вероятностью не больше $\frac{1}{2}$ на входе (x, y)

Заметим, что по определению, $NP = PCP(0, poly(n))$. Это система доказательств, в которой верификатор не подбрасывает монет и читает весь вход доказательства как обычную NP-машину (в этом случае, вероятность того, что x не в языке L, но принимается верификатором равна нулю, а не $\frac{1}{2}$). Описанная ниже, **PCP теорема** показывает, что есть огромный компромисс между параметрами $r(n)$ и $q(n)$. Позволяя верификатору подбрасывать небольшое количество монет, можно значительно уменьшить количество битов запроса, которые требует верификатор.

Теорема 3.1 (PCP)**.** $NP = PCP(log(n), 1)$

Если верификатор должен обнаружить, что утверждение доказывающего является недействительным, проверяя только постоянное число битов, то доказательство должно быть закодировано таким образом, чтобы распространять ошибки по всему доказательству. Это сходится по духу к традиционными целями теории обнаружения ошибок. Полное доказательство этой теоремы также использует следующую композиционную лемму.

Лемма 3.2. Если L лежит одновременно и в $PCP(r_1(n), q_1(n))$ и в $PCP(r_2(n), q_2(n))$, то существуют константы c_1 и c_2 такие что L находится в $PCP(r(n), q(n))$, где $r(n) = r_1(n) + r_2(q_1(n)^{c_1})$ и $q(n) = q_2(q_1(n)^{c_2})$.

Мы дадим представление о системе доказательств $PCP(poly(n), 1)$ для NP-полного языка $3SAT$.

Напомним, что $3SAT$ содержит множество $v_1, ..., v_n$ логических переменных и множество $C_1, ..., C_m$ некоторых условий. Каждое C_j это дизъюнкция трех литералов, где каждый литерал это либо переменная, либо её отрицание. Истинным набором $(a_1, ..., a_n), a_i \in \{0, 1\}$ который делает все условия истинными одновременно, где v_i присвоено значение a_i. Мы будем использовать тот факт, что можно случайно выбрать полином степени 3 $\phi_\tau \in GF(2)[X_1, ..., X_n]$ так, чтобы если $(a_1, ..., a_n)$ - истинный набор, то $\phi_\tau(a_1, ..., a_n) = 0$ с вероятностью 1, иначе $\phi_\tau(a_1, ..., a_n) = 0$ с вероятностью в худшем случае $\frac{1}{2}$. Это возможно с помощью арифметизации каждого C_j. Положительный литерал v_i арифмитизируется как (1-X_i) и негативный литерал $\overline{v_i}$ как X_i, а зачем все три арифмитизированных литерала в условии перемножаются. Например, условие $C_j = v_1 v_2 \overline{v}_3$ арифметизируется как моном $C'_j = (1 - X_1)(1 - X_2)X_3$. Заметим, что C'_j равняется нулю для любого набора, удовлетворяющего C_j. Чтобы выбрать рандомный полином ϕ_τ с описанными свойствами, выбирем τ равномерно из $\{0, 1\}^m$ и пусть $\phi_\tau(X_1, ...X_n) = \sum_{j=1}^{m} \tau_j C'_j$.

Любой вектор $(a_1, ..., a_n)$ в $GF(2)^n$ определяет три линейные функции $A : GF(2)^n \to GF(2)$, $B : GF(2)^{n^2} \to GF(2)$ и $C : GF(2)^{n^3} \to GF(2)$ следующим образом:

$$A(x) = \sum_{i=1}^{n} a_i x_i$$

$$B(y) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_{i,j}$$

$$C(z) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_i a_j a_k z_{i,j,k}$$

Явные таблицы функций для A, B и C имеют экспоненциальный размер от n. Таким образом, индекс записи таблицы может быть записан с помощью $poly(n)$ бит.

Предполагается, что доказывающий запишет таблицы соответствующие утверждению $(a_1, ..., a_n)$, которые, как он считает, удовлетворяют $3SAT$. Имея три таблицы, верификатор V может проверить, что они в самом деле представляют собой три линейные функции, или по крайне мере три функции похожие на линейные. Для этого, V использует следующий метод тестирования: вместо того, чтобы делать запросы к программе для вычисления линейной функции, V просто смотрит на значения функций в таблице. Если таблицы проходят этот тест, V не может быть уверен, что они точно представляют собой линейные функции. Если бы он мог быть уверен в этом, то он бы смог получить любое желаемое значение фукций $A(x), B(y)$ или $C(z)$ просто просто поискав его в таблице. Однако тестер гарантирует, что если таблицы проходят тест, то существуют уникальные линейные функции A, B и C которые очень близки к функциям данным в таблицах. Более того, V может получить достаточно точные значения функций $A(x)$, $B(y)$ или $C(z)$ используя корректирование ошибок. Таким образом, ресурсы, нужные на тестирование линейности и корректирование ровно такие же, как и необходимые для верифицирования в системе доказательств $PCP(poly(n), 1)$: $poly(n)$ рандомных битов необходимо для спецификании рандомных элементов x, y и z в области области определения A, B и C, но необходимо только $O(1)$ вхождений в таблицу[3].

V также может использовать $poly(n)$ рандомных бит и $O(1)$ битов запроса чтобы верифицировать, что линейные функции A, B и C определены разумным множеством таблиц и соотносятся друг с другом правильным образом. Пусть $a' = (a_1, ..., a_n)$, $b' = (b_{1,1}, ..., b_{n,n})$, и $c' = (c_{1,1,1}, ..., c_{n,n,n})$ будут векторами коэффициентов для этих функций. И тогда $a' \cdot (a')^T$ и b' есть матрицы $n \times n$. Обозначим их как M_1 и M_2 соответственно. Пусть τ и s будут случайными векторами из $GF(2)^n$ выбранными равномерного. Если $M_1 \neq M_2$, то $\tau^T M_1 s \neq \tau^T M_2 s$ с вероятностью не меньше $\frac{1}{4}$. По определению, $\tau^T M_1 s = A(\tau) \cdot A(s)$, где $\cdot$ это просто перемножение в $GF(2)$ и $\tau^T M_2 s = B(r \cdot s^T)$. Таким образом, если есть ошибка в отношениях между A и B V может обнаружить её с вероятностью не меньше $\frac{1}{4}$, выбирая два рандомных вектора полиномиальной длины и вычисляя значение функции в трех точках. Эти вычисления требуют только константого числа запросов к таблице. Тест может быть повторен константное число раз для увеличения вероятности обнаружения ошибок. Тест $c' = a' \cdot b'$ аналогичен.

Осталось показать, как V может использовать линейные функции A, B и C чтобы проверить, что утверждение $(a_1, ..., a_n)$, с которого мы начинали, удовлетворяет исходному $3SAT$. Существенным моментом является то, что A, B, C позволяют вычислить любые полиномы степени 3 принадлежащие $GF(2)[X_1, ..., X_n]$ в точке $(a_1, ..., a_n)$. Для любой такой функции f, существует множество индексов $S_1 \subseteq 1, ..., n$, $S_2 \subseteq (1,1), ..., (n,n)$ и $S_3 \subseteq (1,1,1), ..., (n,n,n)$ и константа $\alpha \in GF(2)$ такая чтобы

$$f(a_1, ..., a_n) = \alpha + A(S_1) + B(S_2) + C(S_3)$$

где $A(S_1)$, означает что мы выбрали значения в A, соответствующие множеству S_1 и аналогично для B и C.

Таким образом, система доказательств $PCP(poly(n), 1)$ для $3SAT$ работает следующим образом. Для того, чтобы доказать, что $(v_1, ..., v_n, C_1, ..., C_m)$ подходит, доказывающий берет утверждение $(a_1, ..., a_n)$, которое он хочет доказать, и строит подходящие таблицы функций A, B и C. Верификатор проверяет, что таблицы удовлетворяют всем необходимым свойствам, как описано выше. Если любая из этих проверок терпит неудачу, V выдаёт 0 и останавливается. Если A, B, C имеют все необходимые свойства, то V равномерно выбирает τ из $0, 1^m$ и строит подходящий полином степени три ϕ_τ от утверждения $(a_1, ..., a_n)$ и условий $(C_1, ..., C_n)$. Затем V использует таблицы A, B и C для вычисления $\phi_\tau(a_1, ..., a_n)$

и возвращает 1 тогда и только тогда, когда $\phi_\tau(a_1, ..., a_n) = 0$.

Два случая композиционной леммы дают PCP теорему. Первое, пусть $r_1(n) = r_2(n) = log(n)$ и $q_1(n) = q_2(n) = log(n)^c$. Это значит, что $NP \subseteq PCP(log(n), loglog(n)^{c'})$ для некоторой константы c'. Теперь пусть $r_1(n) = log(n), q_1(n) = loglog(n)^{c'}, r_2(n) = poly(n)$ и $q_2(n) = 1$. Сопостовляя эти два множества параментров мы получаем теорему (Что быть точным, это даёт только одно направление теоремы, а именно $NP \subseteq PCP(log(n), 1)$, обратное включение тривиально следует из определения двух классов.)

Для некоторых потенциальных применений желательно иметь систему доказательств $PCP(log(n), 1)$, в которых доказательств y как можно короче. Например, автоматизированные инструменты для проверки доказательств часто создают доказательства, которые слишком длины, чтобы быть проверены человеком. Техника PCP может предоставить подход для решения этой проблемы. Нужно закодировать автоматически сгенерированное доказательство с помощью теоремы PCP и попросить PCP-верификатора проверить его, используя только постоянное число бит. Так как доказательства, получаемые с помощью автоматизированных инструментов, слишком большие, чтобы хранить или передавать эффективно, важно, чтобы процесс кодирования PCP увеличивал длину доказательства как можно меньше.

Покажем теперь, как PCP теорема используется для доказательства неаппроксимируемости результатов для NP-трудных задач оптимизации. Поскольку аппроксимируемость таких функций является основной проблемой теоретической информатики, эти результаты демонстрируют, что теория кодирования предоставляет достаточно полезные результаты.

Начнем с определения функции $MAX - PCP$, чья область определения состоит из пар (x, L), где L - язык, лежащий в NP, и x является входом размером n, для которого можно определить лежит ли он в L. Язык L представлен описанием системы доказательств $PCP(log(n), 1)$. Заметим, что такая система должна существовать по теореме PCP и что её описание занимает размер $O(1)$. Вопрос лежит ли x в L может быть трансформирован в задачу оптимизации следующим образом. Строки с доказательством в $PCP(log(n), 1)$ системе имеют полиномиальную длину, скажем $s(n)$. Каждый бит строки доказательства y рассматривается как логическая переменная y_i. Проверка в этой системе доказательств подбрасывает последовательность монет, которая имеет длину $c \cdot log(n)$. Таким образом, есть полиномиально много возможных последовательностей подбрасываний монет τ. Пусть $C_{x,\tau}$ будет ограничением на множество переменных $y_1, ..., y_{s(n)}$ равное 1, если верификатора принимает x, если верификатор подбрасывает последовательность τ и равное 0 иначе. Так как верификатор лежит в $PCP(log(n), 1)$, то он читает только O(1) бит доказательства y. Значение $MAX - PCP$ на входе (x, L) это, таким образом, максимум, над $y \in \{0, 1\}^{s(n)}$, среди чисел одновременно удовлетворяющих ограничениям $C_{x,\tau}$.

Теперь мы докажем, что $(\frac{1}{10})$-аппроксимация $MAX-PCP$ лежит среди NP-полных задач. Это следует непосредственно из наличия "разрыва"в вероятностях принятия при определении вероятностной проверяемой системы доказательств. Если $x \in L$, то $MAX-PCP(x, L) = n^c$, то есть существует некоторое доказательство y, для которого верификатор принимается на всех последовательностях подбрасываний монет τ и, следовательно, все ограничения могут быть выполнены одновременно. С другой стороны, если $x \notin L$, то $MAX - PCP(x, L) \leq 0.5n^c$, потому что, для всех строк доказательств y, верификатор принимает не больше $\frac{1}{2}$ подбрасываний монет τ и значит не больше $\frac{1}{2}$ ограничений одновременно удовлетворены. Таким образом, алгоритм $\frac{1}{10}$-аппроксимации для $MAX-PCP$ всегда должен вернуть значение не меньшее $(1/1.1)n^c$ или не большее $0.55n^c$, что позволяет нам различать случаи $x \in L$ и $x \notin L$. Поскольку L это произвольный язык из NP это означает, что $\frac{1}{10}$-аппроксимация $MAX - PCP$ является NP-трудной.

В заключении отметим, что PCP теорема использовалась для получения результата невозможности аппроксимации для многих естественных функций оптимизации, включая хроматического числа, кликового числа, $MAX - 3SAT$ и других

Список литературы

[1] L.Babai, L.Fortnow and C.Lund, Nondeterministic Exponential Time has Two-prover Interactive Protocols, 1991

[2] M. Ben-Or, S.Goldwasser, J.Killian and A.Widgerson, Multiprover Interactive Proof Systems: How to Remove Intractability Assumptions, 1988

[3] Joan Feigenbaum, The Use of Coding Theory in Computational Complexity