Полная задача в классах **AvgBPP** и **HeurBPP**

Д.М. Ицыксон *

10 мая 2008 г.

Аннотация

В работе строится распределенная задача, которая полна в классах **AvgBPP** и **HeurBPP** с полиномиально моделируемыми распределениями относительно детерминированных сведений по Тьюрингу. Класс **AvgBPP** состоит из распределенных задач, которые могут быть решены полиномиальными в среднем по времени вероятностными алгоритмами с двусторонней ошибкой. Класс **HeurBPP** состоит из задач, решаемых вероятностными машинами Тьюринга с двусторонней ошибкой за время $poly(\frac{n}{\delta})$ на всех входах длины n за исключением множества вероятности δ. Поскольку сведения детерминированные, существование полиномиального в среднем (эвристического) детерминированного алгоритма для построенной задачи будет означать, что $\mathbf{AvgP} = \mathbf{AvgBPP}$ ($\mathbf{HeurP} = \mathbf{HeurBPP}$).

Стоит отметить, что хотя легко построить полную promise-задачу в классе **promise-BPP** [Mil01], неизвестно, содержит ли класс **BPP** полные языки.

Также в работе сравниваются вышеперечисленные классы с их классическими аналогами и доказывается, что включения строгие.

1 Введение

Существование полных задач и теоремы об иерархии по времени (и памяти) являются основными структурными свойствами классов сложности. Классы сложности обычно определяются при помощи моделей вычислений. Например, класс **P** определяется полиномиальными по времени детерминированными машинами Тьюринга, **NP** — недетерминированными, а **BPP** — вероятностными машинами с ограниченной двусторонней ошибкой. Теорема об иерархии по времени утверждает, что данная модель вычислений может решить больше задач за большее время. Полная задача — это “самая сложная” задача в классе, все остальные задачи сводятся к ней.

И доказательство иерархий, и построение полных задач обычно требует эффективного перечисления корректных машин в соответствующей модели вычислений. Неизвестно, есть ли иерархия по времени в классе **BPP**, и есть ли в нем полные задачи относительно детерминированных сведений. Основное препятствие — это отсутствие эффективного перечисления вероятностных машин Тьюринга с ограниченной двусторонней ошибкой. Барак показал в [Bar02], что существование полной задачи в классе **BPP** влечет теорему о иерархии по времени для класса **BPP**. Однако существует такой оракул A, что в $\mathbf{BPP}^A$ нет полных задач [HH86]. Заметим, что если $\mathbf{P} = \mathbf{BPP}$, то **BPP** содержит полную задачу, так как в **P** есть полные задачи. Лучший известный результат в области иерархий по времени суперполиномиальный: $\mathbf{BPTime}[n^{\log n}] \subsetneq \mathbf{BPTime}[2^{n^\epsilon}]$ [KV87], однако, мы не можем даже показать, что

*Санкт-Петербургское отделение Математического института им. В.А. Стеклова РАН. E-mail: `dmitrits@pdmi.ras.ru`.

$\mathbf{BPTime}[n] \subsetneq \mathbf{BPTime}[n^{100\log n}]$. Похожие результаты о вероятностных классах сложности включают иерархии для классов с 1 битом подсказки, зависящей только от длины входа: $\mathbf{BPP}/1, \mathbf{ZPP}/1, \mathbf{MA}/1$.

Эвристические алгоритмы могут давать неверный ответ на маленькой доле входов; мы предполагаем, что задано некоторое распределение на входах. Иерархия по времени для класса $\mathbf{Heur}_{\frac{1}{n^c}}\mathbf{BPP}$ (с равномерным распределением) была доказана в работах [FS04, Per07]. Полная криптосистема была построена с использованием похожих идей в [HKN$^+$05] (см. также [GHP06]). Такая криптосистема возможна, когда декодирующий алгоритм может ошибаться с некоторой маленькой вероятностью.

Класс **HeurBPP** [BT06, Imp95] состоит из языков, решаемых на вероятностных машинах Тьюринга ограниченными по времени $poly(\frac{n}{\delta})$ на входах длины n (кроме множества вероятности δ, когда δ дано алгоритму на вход). Доля неправильных ответов может быть понижена увеличением времени работы. Рассмотрение класса **HeurBPP** было мотивировано классом **AvgBPP** [BT06, Imp95], который представляет собой класс задач, решаемых за полиномиальное в среднем время вероятностными алгоритмами с ограниченной ошибкой. Основное отличие между **HeurBPP** и **AvgBPP** в том, что **HeurBPP**-алгоритмы могут давать неверные ответы, а **AvgBPP**-алгоритмы обязаны явно отвечать “не знаю”. В этой работе мы ограничиваемся рассмотрением только полиномиально моделируемых (PSamplable) распределений.

Основным результатом работы является конструкция полной распределенной задачи с полиномиально моделируемыми распределениями в классах **AvgBPP** и **HeurBPP**. Из нашей конструкции следует, что если эта задача содержится в **AvgP** (и даже $\mathbf{Avg}_{\frac{1}{n^c}}\mathbf{P}$), то $\mathbf{AvgP} = \mathbf{AvgBPP}$ (и аналогичное утверждение про **HeurBPP**).

То, что мы рассматриваем полиномиально моделируемые распределения важно для нашей конструкции, и доказательство не работает для более простых распределений (например, полиномиально вычислимых и просто равномерных). Основная проблема с равномерными распределениями следующая: сведение требует обычно удлинения размера входа, что экспоненциально уменьшает вероятность и нарушает условие доминирования.

Кроме того, мы сравниваем классы $\mathbf{AvgP}, \mathbf{AvgBPP}, \mathbf{HeurP}, \mathbf{HeurBPP}$ с классами сложности в наихудшем случае и показываем:

- $\mathbf{P} \subsetneq \mathbf{AvgP} \subseteq \mathbf{HeurP} \subsetneq \mathbf{EXP}$;
- $\mathbf{BPP} \subsetneq \mathbf{AvgBPP} \subseteq \mathbf{HeurBPP} \subsetneq \mathbf{BPEXP}$.

Организация работы. В разделе 2 строго определяются используемые понятия. В разделе 3 мы сравниваем классы сложности в среднем и наихудшем случае, в разделе 4 мы строим полные задачи в классах **AvgBPP** и **HeurBPP**.

2 Предварительные све́дения

Мы ограничимся рассмотрением алфавита из двух символов и будем обозначать множество слов в нем через $\{0,1\}^*$. Языком мы называем любое подмножество слов из $\{0,1\}^*$. Мы будем отождествлять язык и характеристическую функцию языка: $x \in L \iff L(x) = 1$.

Напомним, что класс сложности **P** состоит из языков L, для которых существует такая полиномиальная по времени детерминированная машина Тьюринга M, что $\forall x L(x) = M(x)$. Класс сложности **BPP** состоит из языков L, для которых существует такая полиномиальная по времени вероятностная машина Тьюринга M, что $\forall x \Pr\{M(x) = L(x)\} \geq \frac{3}{4}$. Аналогично заменой ограничений по времени с полиномиального на экспоненциальный ($2^{n^{O(1)}}$) получаются определения классов **EXP** и **BPEXP**.

В теории сложности в среднем случае вместе с языком мы будем рассматривать распределение на его входах.

Определение 2.1. Ансамблем распределений называется семейство $D = \{D_n\}_{n=1}^{\infty}$, где $D_n\colon \{0,1\}^n \to \mathbb{R}_+$ — распределение входов длины n (т.е., $\sum_{a\in\{0,1\}^n} D_n(a) = 1$).

Определение 2.2. Распределенной задачей называется пара (L, D), где $L \subset \{0,1\}^*$ — язык, а D — ансамбль распределений.

Мы ограничимся рассмотрением только полиномиально моделируемых распределений:

Определение 2.3. Распределение D называется полиномиально моделируемым (PSamplable), если существует такой вероятностный алгоритм (сэмплер) S, что на входе 1^n его выходы распределены согласно D_n. Мы будем обозначать **PSamp** множество всех полиномиально моделируемых распределений.

Равномерное распределение будем обозначать U. Очевидно, что $U \in$ **PSamp**.

2.1 Классы полиномиальных в среднем алгоритмов

Мы будем считать, что есть специальный символ $\perp$, который алгоритм может выдать в качестве ответа "не знаю".

Определение 2.4 ([BT06, определение 2.8]). **AvgP** — это класс распределенных задач (L, D), для которых существует алгоритм с двумя параметрами (x, δ), такой что

- время работы A ограничено полиномом относительно $\frac{|x|}{\delta}$;
- для всех x, для которых $D(x) > 0$, выполняется $A(x, \delta) \in \{L(x), \perp\}$;
- для всех n выполняется $\Pr_{x\leftarrow D_n}\{A(x) = \perp\} < \delta$.

Определение 2.5 ([BT06, определение 2.13]). Распределенная задача (L, D) содержится в классе **AvgBPP**, если существует такой алгоритм A с двумя параметрами (x, δ), что

- время работы A ограничено полиномом относительно $\frac{|x|}{\delta}$;
- для всех x для которых $D(x) > 0$ выполняется $\Pr\{A(x, \delta) \notin \{L(x), \perp\}\} < \frac{1}{4}$;
- для всех n выполняется $\Pr_{x\leftarrow D_n}\{\Pr\{A(x, \delta) = \perp\} \geq \frac{1}{4}\} < \delta$.

В определениях 2.4 и 2.5 можно было долю входов, на которой алгоритм может выдавать ответ $\perp$, за счет увеличения времени работы. Можно определить класс без возможности понижать долю входов, на которой задача не решается:

Определение 2.6 ([BT06, определение 2.9]). Для функции $\delta(n)$ определим класс $\mathbf{Avg}_{\delta(n)}\mathbf{P}$. Распределенная задача (L, D) содержится в классе $\mathbf{Avg}_{\delta(n)}\mathbf{P}$, если существует такой алгоритм A, что

- время работы A ограничено полиномом;
- для всех x, для которых $D(x) > 0$, выполняется $A(x) \in \{L(x), \perp\}$;
- для всех n выполняется $\Pr_{x\leftarrow D_n}\{A(x) = \perp\} < \delta(n)$.

Аналогично определяется и класс $\mathbf{Avg}_{\delta(n)}\mathbf{BPP}$.

2.2 Классы эвристических полиномиальных в среднем алгоритмов

Эвристические алгоритмы могут давать неверный ответ (вместо явного ответа $\perp$) на маленькой доле входов.

Определение 2.7 ([BT06, определение 2.10]). Распределенная задача (L, D) лежит в классе **HeurP**, если существует детерминированный алгоритм $A(x, \delta)$,

- полиномиальный по времени относительно $\frac{|x|}{\delta}$;
- такой, что для всех n выполняется $\Pr_{x \leftarrow D_n}\{A(x, \delta) \neq L(x)\} < \delta$.

Определение 2.8 ([BT06, определение 2.15]). Распределенная задача (L, D) лежит в классе **HeurBPP**, если существует вероятностный алгоритм $\mathcal{A}(x, \delta)$,

- полиномиальный по времени относительно $\frac{|x|}{\delta}$;
- такой, что для всех n выполняется $\Pr_{x \leftarrow D_n}\{\Pr\{\mathcal{A}(x, \delta) \neq L(x)\} \geq \frac{1}{4}\} < \delta$.

Определение 2.9 ([BT06, определение 2.11]). Распределенная задача (L, D) лежит в классе $\mathbf{Heur}_{\delta(n)}\mathbf{P}$, если существует такой детерминированный полиномиальный по времени алгоритм $A(x)$, что для всех n выполняется $\Pr_{x \leftarrow D_n}\{A(x, \delta) \neq L(x)\} < \delta(n)$.

Замечание 2.1. Заметим, что $\mathbf{Heur}_{\frac{1}{n^c}}\mathbf{P} \subseteq \mathbf{HeurP}$ (и $\mathbf{Avg}_{\frac{1}{n^c}}\mathbf{P} \subseteq \mathbf{AvgP}$).

2.3 Свед́ения

Для построения полных задач нам понадобятся детерминированные сведения для распределенных задач. Сведения в случае эвристических и неэвристических алгоритмов будут немного отличаться.

Определение 2.10 (ср. [BDCGL92]). Распределенная задача (L, D) сводится к задаче (L', D'), если существует такой алгоритм с оракулом $T^{L'}(x, \delta)$, что выполняются следующие свойства:

1. (Эффективность) Время работы $T^{L'}(x, \delta)$ полиномиально относительно $\frac{|x|}{\delta}$.
2. (Корректность) Для всех тахих x, что $D(x) > 0$, $T^{L'}(x, \delta) \in \{L(x), \perp\}$. Для всех n выполняется $\Pr_{x \leftarrow D_n}\{T^{L'}(x, \delta) = \perp\} < \delta$.
3. (Доминирование) Существует такой полином $p(n)$, что для всех n и δ $\sum_{x \in D_n} Ask_{T,\delta}(x, y) D_n(x) \leq p(\frac{n}{\delta}) D'(y)$, где $Ask_{T,\delta}(x, y) = 1$ если $x \notin E_n$ и $T^{L'}(x, \delta)$ запрашивает оракул y, $Ask_{T,\delta}(x, y) = 0$ иначе, где $E_n \subseteq \{0, 1\}^n$ — некоторое подмножество маленького размера: $D_n(E_n) \leq \delta$.

Будем говорить, что распределенная задача (L, D) *эвристически* сводится к (L', D'), если условие корректности формулируется следующим образом: $\Pr_{x \leftarrow D_n}\{T^{L'}(x, \delta) \neq L(x)\} < \delta$.

Следующие леммы показывают, что такие определения сведений действительно имеют право на существование:

Лемма 2.1. (1) Если (L, D) сводится к (L', D') и $(L', D') \in \mathbf{AvgP}$, то $(L, D) \in \mathbf{AvgP}$.
(2) Если (L, D) эвристически сводится к (L', D') и $(L', D') \in \mathbf{HeurP}$, то $(L, D) \in \mathbf{HeurP}$

Доказательство. (1) Пусть (L', D') решается алгоритмом $A'(x,\delta)$ в **AvgP** и $T^{L'}(x,\delta)$ — сведение задачи (L, D) к (L', D'). Пусть время работы $A'(x,\delta)$ ограничено полиномом $q(\frac{|x|}{\delta})$, а количество длин запросов $T^{L'}(x,\delta)$ к оракулу ограничено полиномом $f(\frac{|x|}{\delta})$, пусть длины запросов для входов длины n: $k_1, k_2, \ldots, k_{f(\frac{n}{\delta})}$. Мы определяем алгоритм $A(x,\delta)$, который запускает $T^{L'}(x,\frac{\delta}{3})$ и моделирует $A'(y,\epsilon(x,y))$ вместо запросов y к оракулу, где $\epsilon(n) = \frac{\delta}{3p(\frac{n}{\delta})f(\frac{n}{\delta})}$. Если оракул отвечает $\perp$, то и $A(x,\delta)$ отвечает $\perp$. Вероятность ответа $\perp$ получившегося алгоритма можно оценить так:

$$\Pr_{x\leftarrow D_n}\{A(x,\delta)=\perp\} \leq$$

$$\Pr_{x\leftarrow D_n}\{T^{L'}(x,\frac{\delta}{3})=\perp\} + D_n(E_n) + \sum_{x\in\{0,1\}^n}\sum_{\substack{y\in\{0,1\}^* \\ A'(y,\epsilon(n))=\perp}} Ask_{T,\delta}(x,y)D_n(x) \leq$$

$$\frac{\delta}{3}+\frac{\delta}{3}+\sum_{x\in\{0,1\}^n}\sum_{i=1}^{f(\frac{n}{\delta})}\sum_{\substack{y\in\{0,1\}^{k_i} \\ A'(y,\epsilon(n))=\perp}} Ask_{T,\delta}(x,y)D_n(x) \overset{\text{(Доминирование)}}{\leq}$$

$$\frac{\delta}{2}+\sum_{i=1}^{f(\frac{n}{\delta})}\sum_{\substack{y\in\{0,1\}^{k_i} \\ A'(y,\epsilon(n))=\perp}} p(\frac{n}{\delta})D'(y) = \frac{2\delta}{3}+\sum_{i=1}^{f(\frac{n}{\delta})} p(\frac{n}{\delta})\Pr_{y\leftarrow D'_{k_i}}\{A'(y,\epsilon(n))=\perp\} <$$

$$\frac{2\delta}{3}+\sum_{i=1}^{f(\frac{n}{\delta})} p(\frac{n}{\delta})\epsilon(n) = \frac{2\delta}{3}+f(\frac{n}{\delta})\frac{\delta}{3f(\frac{n}{\delta})} = \delta.$$

(2) Доказательство аналогично пункту (1). □

Следствие 2.1 (из доказательства леммы 2.1). Пусть (L, D) (эвристически) сводится к (L', D') с помощью сведения $T^{L'}(x,\delta)$, количество длин запросов $T^{L'}(x,\delta)$ к оракулу ограничено полиномом $f(\frac{|x|}{\delta})$, $p(\frac{|x|}{\delta})$ — это полином из условия доминирования. Все запросы y к оракулу удовлетворяют неравенству: $|y| \geq \lceil(\frac{1}{\epsilon(|x|)})^{\frac{1}{c}}\rceil$, где $\epsilon(n) = \frac{\delta}{3p(\frac{n}{\delta})f(\frac{n}{\delta})}$. (1) Если $(L', D') \in \mathbf{Avg}_{\frac{1}{n^c}}\mathbf{P}$, то $(L, D) \in$ **AvgP**. (2) Если $(L', D') \in \mathbf{Heur}_{\frac{1}{n^c}}\mathbf{P}$, то $(L, D) \in$ **HeurP**.

Доказательство. Доказательство отличается от доказательства леммы 2.1 тем, что задача (L', D') решается алгоритмом $A'(y)$. Для всех запросов y, которые делает $T^{L'}(x,\delta)$, выполняется неравенство $\Pr_{y\in D'_{|y|}}\{A'(y)=\perp\} < \frac{1}{|y|^c} \leq \epsilon(|x|)$. □

2.4 Понижение констант в AvgBPP

Покажем, что константа $\frac{1}{4}$ в определении 2.5 может быть экспоненциально понижена.

Предложение 2.1 (оценка Чернова-Хоефдинга). Для независимых и одинаково распределенных случайных величин $X_1, X_2, \ldots, X_N$, таких что $X_i \in [0,1]$ и $E[X_i] = \mu$, выполняется $\Pr\{|\frac{\sum_{i=1}^N X_i}{N} - \mu| \geq \epsilon\} \leq 2e^{-2\epsilon^2 n}$.

Лемма 2.2 ([BT06]). Для того, чтобы распределенная задача (L, D) содержалась в классе **AvgBPP** необходимо и достаточно, чтобы существовал такой алгоритм B с двумя параметрами (x,δ), что

- время работы B ограничено полиномом относительно $\frac{|x|}{\delta}$;
- для всех x, для которых $D(x) > 0$, выполняется $\Pr\{B(x) \notin \{L(x), \perp\}\} \le 2^{-\Omega(n)}$;
- для всех n выполняется $\Pr_{x \leftarrow D_n}\{\Pr\{B(x,\delta) = \perp\} < 2^{-\Omega(n)}\} \ge 1 - \delta$.

Доказательство. Достаточность очевидна. Докажем необходимость.

Пусть (L, D) содержится в **AvgBPP** и решается алгоритмом $A(x, \delta)$ согласно определению 2.5. Опишем алгоритм B следующим образом: повторим алгоритм $A(x, \delta)$ $n = |x|$ раз, если хотя бы $\frac{n}{3}$ ответов равняются $\perp$, то выдать $\perp$. Если менее $\frac{n}{3}$ ответов равняются $\perp$, то выдать наиболее часто встречаемый ответ. Разобьем все строки длины n на три множества: $X_1 = \{x \in \{0,1\}^n | \Pr\{A(x, \delta) = \perp\} < \frac{1}{4}\}$, $X_2 = \{x \in \{0,1\}^n | \frac{1}{4} \le \Pr\{A(x, \delta) = \perp\} \le \frac{1}{3} + \frac{1}{10}\}$ и $X_3 = \{x \in \{0,1\}^n | \Pr\{A(x, \delta) = \perp\} > \frac{1}{3} + \frac{1}{10}\}$.

Из оценки Чернова следует, что если $x \in X_1$, то с вероятностью $1 - 2^{-\Omega(n)}$ менее $\frac{1}{3}$ ответов будут совпадать с $\perp$. В этом случае $\Pr\{B(x, \delta) = L(x)\} \ge 1 - 2^{-\Omega(n)}$. Если $x \in X_3$, то $\Pr\{B(x, \delta) = \perp\} \ge 1 - 2^{-\Omega(n)}$. Если $x \in X_2$, то $\Pr\{B(x, \delta) = 1 - L(x)\} \le 1 - 2^{-\Omega(n)}$. Из определения 2.5 следует, что $D(X_2 \cup X_3) < \delta$, а при $x \in X_1$ выполняется $\Pr\{B(x, \delta) = L(x)\} \ge 1 - 2^{-\Omega(n)}$. Значит, $\Pr_{x \leftarrow D_n}\{\Pr\{B(x, \delta) = \perp\} < 2^{-\Omega(n)}\} \ge 1 - \delta$.

□

3 Сложность в среднем и в наихудшем случае

Легко показать, что **AvgP** $\subseteq$ **HeurP**. Действительно, достаточно модифицировать алгоритм для **AvgP** следующим образом: заменить ответы $\perp$ на 0. Является ли это включение строгим — открытый вопрос [Imp95].

Аналогично с помощью леммы 2.2 получаем **AvgBPP** $\subseteq$ **HeurBPP**.

Замечание 3.1. Если мы к классическому классу сложности добавляем распределение на входах, то имеется в виду, что задача должна решаться в рамках класса на всех входах, вероятность которых положительна.

Теорема 3.1. Выполняются следующие соотношения:

1. $(\mathbf{P}, U) \subsetneq (\mathbf{AvgP}, U) \subseteq (\mathbf{HeurP}, U)$;
2. $(\mathbf{HeurP}, \mathbf{PSamp}) \subseteq (\mathbf{EXP}, \mathbf{PSamp})$;
3. Существует такой язык $L_{EXP} \in \mathbf{EXP}$, что для любого распределения $D \in \mathbf{PSamp}$ задача (L, D) не содержится в классе $(\mathbf{HeurP}, \mathbf{PSamp})$.

Доказательство. 1. Перечислим все детерминированные машины Тьюринга, работающие время $O(2^n)$: M_i (для этого достаточно перечислять пары (M, c), где M — машина Тьюринга, которую останавливают через $c2^n$ шагов, $c \in \mathbb{N}$). Рассмотрим язык $L_P = \{0^i | M_i(0^i) = 0\}$. Пусть язык L_P распознается полиномиальной по времени машиной Тьюринга, она имеет номер в списке M_k. Пусть $0^k \in L_P$, тогда $M_k(0^k) = 1$, так как M_k распознает L_P, с другой стороны по определению L_P должно быть $M_k(0^k) = 0$, противоречие. Пусть $0^k \notin L_P$, тогда поскольку M_k распознает язык L_p имеем $M_k(0^k) = 0$, а это означает, что $0^k \in L_P$, противоречие. Значит, $(L_P, U) \notin (\mathbf{P}, U)$.

Покажем, что $(L_p, U) \in \mathbf{AvgP}$. Алгоритм $A(x, \delta)$ при $x \ne 0^n$ выдает 0. При $x = 0^n$ и $\delta > \frac{1}{2^n}$ выдает $\perp$, при $\delta \le \frac{1}{2^n}$, моделирует машину M_n на входе 0^n и обращает ее результат. Время работы алгоритма $A(x,\delta)$ ограничено числом $O(\frac{x}{\delta^2})$ (при $\delta \le 2^{-n}$, алгоритм $A(x,\delta)$ может работать 2^{2n} шагов).

2. Пусть задача $(L,D) \in (\mathbf{HeurP}, \mathbf{PSamp})$ распознается алгоритмом $A(x,\delta)$, распределение D генерируется самплером S, время работы которого ограничено полиномом $q(n)$. Заметим, что минимальная положительная вероятность входа длины n согласно распределению D не меньше, чем $2^{-q(n)}$. Тогда алгоритм $A(x, \frac{1}{2^{q(|x|)+1}})$ работает экспоненциальное от $|x|$ время и безошибочно распознает L на входах, вероятность которых согласно D положительна.

3. Снова рассмотрим перечисление всех корректных машин, работающих время $O(2^n)$: M_i. И язык $L_{EXP} = \{x | M_{|x|}(x) = 0\}$. Этот язык распознается в **EXP** простым моделированием. Пусть этот язык с каким-то распределением D распознается в **HeurP** с помощью алгоритма $A(x,\delta)$. Подставим в этот алгоритм $\delta = \frac{1}{10}$. Пусть у получившегося алгоритма номер в нашем перечислении k. Тогда на входах длины k $A(x, \frac{1}{10})$ не дает ни одного правильного ответа, а должен давать правильные ответы на множестве входов вероятности хотя бы 0.9. Противоречие. □

Теперь докажем аналогичную теорему для вероятностных классов:

Теорема 3.2. Выполняются следующие соотношения

1. $(\mathbf{BPP}, U) \subsetneq (\mathbf{AvgBPP}, U) \subseteq (\mathbf{HeurBPP}, U)$;
2. $(\mathbf{HeurBPP}, \mathbf{PSamp}) \subseteq (\mathbf{BPEXP}, \mathbf{PSamp})$;
3. Существует язык $L \in \mathbf{BPPEXP}$, что для любого распределения $D \in \mathbf{PSamp}$ задача (L,D) не содержится в классе $(\mathbf{HeurBPP}, \mathbf{PSamp})$.

Доказательство. 1. Чтобы доказать $(\mathbf{BPP}, U) \subsetneq (\mathbf{AvgBPP}, U)$, достаточно показать, что существует унарный язык (унарный язык — это подмножество $\{0\}^*$), который различает классы **BPP** и $\mathbf{BPTime}[2^n]$. Приставка **u-** к классу означает, что в классе оставили только унарные языки. Доказательство будет напоминать доказательство $\mathbf{BPP} \neq \mathbf{BPTime}[2^n]$ [1] из [KV87]. Предположим, что $\mathbf{u\text{-}BPP} = \mathbf{u\text{-}BPTime}[2^n]$, тогда $\mathbf{u\text{-}BPP} = \mathbf{u\text{-}BPTime}[n^{\log n}] = \mathbf{u\text{-}BPTime}[2^n]$.

Лемма 3.1 (ср. [KV87, лемма 3]). Пусть функции $f(n), g(n), h(n)$ — конструктивные по времени [2], $f(n), g(n) \ge \log n$, $h(n) \ge n$ — строго возрастающая функция. Тогда $\mathbf{u\text{-}BPTime}[f(n)] \subseteq \mathbf{u\text{-}BPTime}[g(n)]$ влечет $\mathbf{u\text{-}BPTime}[f(h(n))] \subseteq \mathbf{u\text{-}BPTime}[g(h(n))]$

Доказательство. Пусть язык A распознается за время $f(h(n))$ алгоритмом M, припишем к каждому входу паддинг длины $h(n) - n$: $A^{pad} = \{x0^{h(|x|)-|x|} | x \in A\}$. Язык A^{pad} можно распознать за время $O(f(n))$ так: по входу y двоичным поиском найти такой x, что $y = x0^{h(|x|)-|x|}$ (за время $O(\log |y|)$). После этого применить к x алгоритм M, время его работы $O(f(|y|))$. Значит, язык A^{pad} можно распознать за время $O(g(|y|))$, а значит язык A можно распознать за время $O(g(h(n)))$. □

Итак, допустим, что $\mathbf{u\text{-}BPTime}[n^{\log n}] = \mathbf{u\text{-}BPTime}[2^n]$, напишем следующую цепочку включений:

[1] Мы используем обозначения $\mathbf{DTime}[f(n)]$ для класса языков, распознаваемых за время $O(f(n))$ на детерминированных машинах Тьюринга, а $\mathbf{BPTime}[f(n)]$ на вероятностных машинах Тьюринга с двусторонней ограниченной ошибкой.

[2] $f(n)$ конструктивна по времени, если значение $f(n)$ может быть вычислено за $O(f(n))$.

$$\mathbf{u\text{-}DTime}[2^{n^{2\log n}}] \subseteq \mathbf{u\text{-}BPTime}[2^{n^{2\log n}}] \overset{\text{лемма 3.1}}{\subseteq} \mathbf{u\text{-}BPTime}[(n^{2\log n})^{\log(n^{2\log n})}] \subseteq \mathbf{u\text{-}BPTime}[2^n] \subseteq \mathbf{u\text{-}BPTime}[n^{\log n}] \subseteq \mathbf{u\text{-}DTime}[2^{n^{\log n}}]$$

Для получения противоречия достаточно показать, что $\mathbf{u\text{-}DTime}[2^{n^{\log n}}] \subsetneq \mathbf{u\text{-}DTime}[2^{n^{2\log n}}]$. Перечислим все детерминированные машины Тьюринга, работающие время $O(2^{n^{\log n}})$: пусть i-ая машина в этом перечислении — это M_i. Тогда язык $L_{BPP} = \{0^i | M_i(0^i) = 0\}$ разделяет $\mathbf{u\text{-}DTime}[2^{n^{2\log n}}]$ и $\mathbf{uDTime}[2^{n^{\log n}}]$. Кроме того, можно сделать вывод, что либо L_{BPP}, либо его версия с паддингом разделяет $\mathbf{u\text{-}BPTime}[n^{\log n}]$ и $\mathbf{u\text{-}BPTime}[2^n]$.

2. Включение $(\mathbf{HeurBPP}, \mathbf{PSamp}) \subseteq (\mathbf{BPEXP}, \mathbf{PSamp})$ доказывается аналогично детерминированному случаю (см. пункт 2 теоремы 3.1).

3. Пусть L' — унарный язык, который разделяет $\mathbf{u\text{-}BPTime}[n^{\log n}]$ и $\mathbf{u\text{-}BPTime}[2^n]$. Определим язык $L = \{x | 0^{|x|} \in L'\}$. Поскольку $L \in \mathbf{BPTime}[2^n]$, то и $L' \in \mathbf{BPTime}[2^n]$. Пусть $(L, D) \in \mathbf{HeurBPP}$ для некоторого полиномиально моделируемого распределения D. Пусть (L, D) решается с помощью алгоритма $A(x, \delta)$ и распределение D моделируется самплером S. Для получения противоречия покажем, что $L' \in \mathbf{BPP}$. Действительно, рассмотрим такой алгоритм, распознающий L'. Если $x \neq 0^n$, то отвергнуть. В противном случае запустить $S(1^n)$ (результат обозначим за y), к результату применить $A(y, \frac{1}{10})$. Вероятность ошибки данного алгоритма складывается из вероятности попасть в долю $\frac{1}{10}$, где алгоритм A может работать некорректно и из собственной ошибки $\frac{1}{4}$ алгоритма $A(y, \frac{1}{10})$. Итого, ошибка не превосходит $\frac{1}{4} + \frac{1}{10}$. Т.е., $L' \in \mathbf{BPP}$, противоречие.

□

Отметим, что классы $\mathbf{AvgP}, \mathbf{HeurP}, \mathbf{AvgBPP}$ и $\mathbf{HeurBPP}$ не инвариантны относительно смены распределения. В теоремах 3.1 и 3.2 мы построили унарные языки с равномерным распределением, разделяющие $\mathbf{P}$ и $\mathbf{AvgP}$ и $\mathbf{BPP}$ и $\mathbf{AvgBPP}$. Но если мы сосредоточим все распределение на входах вида 0^n, то соответствующие задачи не будут решаться в классах $\mathbf{AvgP}$ ($\mathbf{HeurP}$) и $\mathbf{AvgBPP}$ ($\mathbf{HeurBPP}$).

4 Полная задача

В теории сложности в среднем случае имеет значение кодирование, в частности мы будем использовать, что вход алгоритма — это кортеж, состоящий из нескольких элементов. Чтобы понизить вероятность строки лишь полиномиально, мы можем использовать только логарифмическое число вспомогательных битов. Опишем, как мы будем кодировать кортежи:

Замечание 4.1. Пусть x и y — две битовые строчки. Будем кодировать пару (x, y) как $0^{\lceil \log |x| \rceil} 1 |x|_2 xy$, где $|x|_2$ — длина строки x, записанная в двоичной системе счисления. Легко видеть, что $|(x, y)| = |x| + |y| + 2\lceil \log |x| \rceil + 1$. Тогда m-местный кортеж $z = (x_1, x_2, \dots, x_m)$ можно закодировать так: $(x_1, (x_2, (x_3, \dots (x_{m-1}, x_m) \dots)$. В этом случае: $|z| = \sum_{i=1}^{m} |x_i| + 2\sum_{i=1}^{m-1} \lceil \log |x_i| \rceil + m - 1 < \sum_{i=1}^{m} |x_i| + 2(m-1)\lceil \log(|z| - |x_m|) \rceil + m - 1$.

Мы будем несколько раз применять оценку Чернова (предложение 2.1), для этих целей зафиксируем такое число N_0, что $2e^{-\frac{N_0}{1000}} < 0.001$. Каждый раз, когда мы будем применять оценку Чернова, количество случайных величин будет не менее N_0.

Мы построим задачу (C, R), где C — язык, а R — полиномиально моделируемое распределение. Язык C мы зададим с помощью алгоритма $\mathcal{A}(x, \delta)$, а распределение R — с помощью самплера $\mathcal{R}$. Мы покажем, что задача распределенная задача (C, R) лежит в классе **AvgBPP** (а значит, и в **HeurBPP**) и что задача (C, R) полная в классе **AvgBPP** (точно так же (даже технически проще) доказывается, что задача (C, R) полна в классе **HeurBPP** относительно эвристических сведений).

Рассмотрим вспомогательный алгоритм $\mathcal{B}$:

Алгоритм 4.1. Алгоритм $\mathcal{B}(x, \delta)$:

1. Проверить, что вход — это строка вида $(M, x, 1^m, b)$, где $m > x + N_0$, $b \in \{0, 1\}$. Если нет, то отвергнуть. (Здесь M — это запись машины Тьюринга, x — вход машины Тьюринга, m — количество шагов, которое разрешается сделать машине, b — ответ, который будет выдаваться вместо ответа $\perp$ машины M.)

2.
   - Если $\delta > \frac{1}{2^m}$, то запустить машину M на m шагов $200m^2$ раз. Если хотя бы один из ответов $\{0, 1\}$ встречается хотя бы в 80% случаев, то выдать этот ответ, иначе выдать $\perp$.
   - Если $\delta \le \frac{1}{2^m}$, то перебрать все последовательности случайных чисел, если хотя бы $\frac{1}{4}$ ответов равняется $\perp$, то выдать b, в противном случае выдать наиболее частый ответ из $\{0,1\}$.

3. Выдать $\perp$.

Заметим, что алгоритм $\mathcal{B}(x, \frac{1}{2^{|x|}})$ работает детерминировано и распознает некий язык B.

Определим на основе алгоритма $\mathcal{B}$ алгоритм $\mathcal{A}$, алгоритм A применяет алгоритм B к части входных данных, а остальные входные данные игнорирует:

Алгоритм 4.2. Алгоритм $\mathcal{A}(x, \delta)$:

1. Проверить, что вход — это строка вида $(M, x, 1^m, b, S, 1^s)$, где $b \in \{0, 1\}$. Если нет, то отвергнуть.

2. Выдать результат работы $\mathcal{B}((M, x, 1^m, b), \delta)$.

Поскольку алгоритм $\mathcal{B}(x, \frac{1}{2^{|x|}})$ работает детерминированно, то и $\mathcal{A}(x, \frac{1}{2^n})$ работает детерминировано и распознает некоторый язык; это будет язык C, для которого мы подберем распределение, чтобы получившаяся задача была полна в классе **AvgBPP**. Неформально говоря, если машина M принимает строку x за m шагов, то язык C содержит строку $(M, x, 1^m, b, S, 1^s)$ для всех $b \in 0, 1, S$ и s, если M выдает $\perp$ на входе x за m шагов, то одна из строк $(M, x, 1^m, 0, S, 1^s)$ и $(M, x, 1^m, 1, S, 1^s)$ содержится в языке, другая нет. Если M некорректно ведет себя на входе x, то результат может быть любой, но вероятность таких входов будет мала за сччт правильного выбора распределения R.

Следующая лемма показывает, какому свойству достаточно удовлетворять распределению H, чтобы распределенная задача (C, H) решалась алгоритмом $\mathcal{A}(x, \delta)$ в **AvgBPP**.

Лемма 4.1. Пусть распределение H обладает таким свойством: для каждой машины Тьюринга M, если вероятность наиболее частого из $\{0, 1\}$ ответа M на x за не более, чем m шагов не превосходит 0.85, то $H(M, x, 1^m, b, S, 1^s) \le 2e^{-n^2}$, где $n = |(M, x, 1^m, b, S, 1^s)|$. Тогда $(C, H) \in$ **AvgBPP**.

Доказательство.

- Если $\delta > \frac{1}{2^m}$, то время работы алгоритма $\mathcal{A}$ ограничено $O(n^4)$. Если $\delta \le \frac{1}{2^m}$, то время работы ограничено $O(\frac{n}{\delta^2})$.

- Пусть $\delta > \frac{1}{2^m}$ (иначе алгоритм ведет себя детерминировано и все время выдает правильный ответ). Тогда в случае, когда вероятность наиболее частого ответа из $\{0,1\}$ машины M на входе x за не более, чем m шагов не больше, чем 0.75, то из оценки Чернова $\Pr\{\mathcal{A}((M,x,1^m,b,S,1^s),\delta) =\perp\} \ge 0.99$. Если же вероятность наиболее частого ответа из $\{0,1\}$ более, чем 0.75, то этот ответ и есть $C(M,x,1^m,b,S,1^s)$. В этом случае $\Pr\{\mathcal{A}((M,x,1^m,b,S,1^s),\delta) = 1 - C(M,x,1^m,b,S,1^s)\} < 0.01$.

- Пусть $\delta > \frac{1}{2^m}$ (иначе алгоритм ведет себя детерминировано и не выдает $\perp$). Заметим, что если вероятность самого частого ответа (из $\{0,1\}$) машины M на x за m шагов не более 0.85, то по условию леммы вероятность такого входа не более $2e^{-n^2}$. Суммарная вероятность таких входов не больше $e^{-n^2}2^{n+1} \le 2^{-n} < \delta$ (при $n > N_0$). В противном случае из оценки Чернова $\Pr\{\mathcal{A}((M,x,1^m,b,S,1^s),\delta) =\perp\} < 0.01$.

□

Определим распределение R с помощью самплера $\mathcal{R}$ (это распределение будет участвовать в определении полной задачи).

Алгоритм 4.3. Самплер $\mathcal{R}(1^n)$:

1. Сгенерировать строку длины n. Если она не имеет вид (M,y,r,b,S,σ), где $b \in \{0,1\}$, то выдать сгенерированную строку.

2. Запустить самплер S на входе $1^{|y|}$ на $|\sigma|$ шагов. Результат обозначим за x.

3. Запустить машину M на $|r|$ шагов $200n^2$ раз. Если каждый из ответов $\{0,1\}$ встречается менее, чем в 90% случаев, то выдать 1^n. (Отметим, что по замечанию 4.1 строка 1^n не кодирует кортеж.)

4. Выдать $(M,x,1^{|r|},b,S,1^{|\sigma|})$.

Лемма 4.2. Пусть самплер S соответствует распределению D. Пусть $z = (M,x,1^m,b,S,1^s)$, $n = |z|$. (1) Если машина M на входе x за m шагов выдает один из ответов $\{0,1\}$ с вероятностью хотя бы 0.95, то $R(z) \ge (1 - 2e^{-n^2})D(x)2^{-10\log(n-s)-5} \cdot 2^{-|M|-|S|}$. (2) Если машина M на входе x за m шагов выдает все ответы с вероятностью не более 0.85, то $R(z) \le 2e^{-n^2}$.

Доказательство. (1) С вероятностью хотя бы $2^{-10\log(n-|\sigma|)-5} \cdot 2^{-|M|-|S|}$ самплер $\mathcal{R}$ на 1-ом шаге сгенерирует строку (M,y,r,b,S,σ) с $|y| = |x|, |r| = m, |\sigma| = s, b \in \{0,1\}$. (Из-за кодирования кортежей по замечанию 4.1 не более, чем $10\log(n-|\sigma|)-5$ битов уйдет на фиксацию размеров элементов кортежей). С вероятностью $D(x)$ на шаге 2 работы самплера $\mathcal{R}$ самплер S выдаст x. Из оценки Чернова получаем, что на шаге 3 самплер $\mathcal{R}$ пройдет тест с вероятностью хотя бы $(1 - 2e^{-n^2})$.

(2) Из оценки Чернова тест на шаге 3 самплера $\mathcal{R}$ будет пройден с вероятностью не более $2e^{-n^2}$. □

Пункт (1) леммы 4.2 утверждает, что построенное распределение R удовлетворяет достаточному условию из леммы 4.1.

Теорема 4.1. $(C,R) \in \mathbf{AvgBPP}$.

Доказательство. Теорема следует из пункта (2) леммы 4.2 и леммы 4.1. $\square$

Замечание 4.2. Пусть машина M имеет два входа: строка x и рациональное число $\delta \in (0,1)$. Пусть M_δ — это машина Тьюринга, которая моделирует работу M с подставленным значением второго параметра $\frac{1}{\lceil\frac{1}{\delta}\rceil}$. Мы можем закодировать M_δ как пару $(M, \lceil\frac{1}{\delta}\rceil)$, где $\lceil\frac{1}{\delta}\rceil$ записана в двоичной системе счисления. По замечанию 4.1 $|(M, \lceil\frac{1}{\delta}\rceil)| = |M| + \lceil\log\lceil\frac{1}{\delta}\rceil\rceil + 2\lceil\log|M|\rceil + 1$, и поэтому $2^{|M_\delta|} \le 2^{|M|+3}M^2(\frac{1}{\delta}+1)$.

Теорема 4.2. (C,R) — полная задача в классе $(\mathbf{AvgBPP}, \mathbf{PSamp})$.

Доказательство. Рассмотрим задачу (L,D) из $\mathbf{AvgBPP}$. Она решается какой-то машиной M_δ за время, ограниченное $g(\frac{|x|}{\delta})$. (Мы считаем, что обе константы в определении 2.5 понижены с помощью леммы 2.2 до 0.01.) Пусть распределение D генерируется с помощью самплера S, время работы которого ограничено полиномом $q(n)$.

Опишем сведение в терминах определения 2.10. Сведение $T^C(x,\delta)$ делает 2 запроса оракулу: $z_0 = (M_\delta, x, 1^{g(\frac{|x|}{\delta})+N_0}, 0, S, 1^{q(|x|)})$ и $z_1 = (M_\delta, x, 1^{g(\frac{|x|}{\delta})+N_0}, 1, S, 1^{q(|x|)})$. Если ответы оракула разные, то выдать $\perp$, иначе выдать ответ оракула. Проверим все свойства сведения.

1. Эффективность следует из того, что строки z_0 и z_1 имеет полиномиальную относительно $\frac{|x|}{\delta}$ длину.

2. Корректность следует из того, что если $C(z_0) = C(z_1)$, то $\Pr\{M_\delta(x) = \perp\} < \frac{1}{4}$. В этом случае по определению 2.5 $\Pr\{M_\delta(x) \in \{L(x), \perp\}\} \ge 0.99$, значит $\Pr\{M_\delta(x) = L(x)\} \ge 0.74$, поскольку $C(z_0)$ — это наиболее частый ответ M_δ на x за не более, чем $g(\frac{x}{\delta})$ шагов, то $C(z_0) = L(x)$.

3. Доминирование. Зафиксируем n, доля входов, на которой M_δ дает ответ $\perp$ с вероятностью хотя бы 0.01, меньше δ. Обозначим все эти входы за E_n (для всех $x \in E_n$ и всех y выполняется $Ask_{T,\delta}(x,y) = 0$, т.е., это множество, которое не участвует в выполнении условия доминирования). При данном δ по запросу к оракулу можно однозначно определить x. На входах из $\{0,1\}^n \setminus E_n$ вероятность наиболее частого ответа M_δ на x не менее 0.98, и этот ответ отличен от $\perp$. По пункту (1) леммы 4.2 $D(M_\delta, x, 1^{g(\frac{|x|}{\delta})+N_0}, b, S, 1^{q(|x|)}) \ge 0.99D(x)2^{-5\log(n'-q(x))-10}\cdot 2^{-|M_\delta|-|S|}$, где $n' = |(M_\delta, x, 1^{g(\frac{|x|}{\delta})+N_0}, b, S, 1^{q(|x|)})|$. Условие доминирования следует из того, что $|S|$ — константа, а размер $|M_\delta|$ зависит логарифмически от $\frac{1}{\delta}$ по замечанию 4.2.

$\square$

Следствие 4.1. Если $(C,R) \in \mathbf{AvgP}$, то $(\mathbf{AvgP}, \mathbf{PSamp}) = (\mathbf{AvgBPP}, \mathbf{PSamp})$.

Доказательство. Следует из теоремы 4.2 и леммы 2.1. $\square$

Теорема 4.3. Если $(C,R) \in \mathbf{Avg}_{\frac{1}{n^c}}\mathbf{P}$, то $(\mathbf{AvgP}, \mathbf{PSamp}) = (\mathbf{AvgBPP}, \mathbf{PSamp})$.

Доказательство. Доказательство повторяет доказательство теоремы 4.2. Мы модифицируем его, чтобы воспользоваться следствием 2.1. Для этого мы искусственно удлиним запросы к оракулу. Пусть в терминах доказательства теоремы 4.2 $k = |(M_\delta, x, 1^{g(\frac{|x|}{\delta})+N_0}, 0, S, 1)| - 1$. Пусть $p(k,\delta) = \frac{1}{0.99}D(x)2^{5\log(k)+|M_\delta|+|S|+9}$, $p(k,\delta)$ ограничено полиномом относительно $\frac{|x|}{\delta}$. Удлиним вход оракула, добавив $\lceil(\frac{1}{\epsilon(n)})^{\frac{1}{c}}\rceil$ единиц ко времени работы самплера S, где

$\epsilon(n) = \frac{\delta}{3p(k,\delta)}$. Новые запросы будут такие: $z_0 = (M_\delta, x, 1^{p(\frac{x}{\delta})+N_0}, 0, S, 1^{q(|x|)+\lceil(\frac{1}{\epsilon(n)})^{\frac{1}{c}}\rceil})$, $z_1 = (M_\delta, x, 1^{p(\frac{x}{\delta})+N_0}, 1, S, 1^{q(|x|)+\lceil(\frac{1}{\epsilon(n)})^{\frac{1}{c}}\rceil})$.

Чтобы воспользоваться следствием 2.1 осталось заметить, что для данной длины входа (и δ) запросы к оракулу имеют одинаковую длину и полином доминирования $p(k,\delta)$ не зависит от длины последнего элемента кортежа (т.е., он такой же, как и в теореме 4.2). □

Следствие 4.2. Если $(\mathbf{AvgBPP}, \mathbf{PSamp}) \subseteq (\mathbf{Avg}_{\frac{1}{n^c}}\mathbf{P}, \mathbf{PSamp})$, то $(\mathbf{AvgP}, \mathbf{PSamp}) = (\mathbf{AvgBPP}, \mathbf{PSamp})$.

Теорема 4.4. 1. Задача (C, R) полна в классе $(\mathbf{HeurBPP}, \mathbf{PSamp})$ относительно эвристических сведений.

2. Если $(C, R) \in \mathbf{HeurP}$, то $(\mathbf{HeurP}, \mathbf{PSamp}) = (\mathbf{HeurBPP}, \mathbf{PSamp})$

3. Если $(C, R) \in \mathbf{Heur}_{\frac{1}{n^c}}\mathbf{BPP}$, то $(\mathbf{HeurP}, \mathbf{PSamp}) = (\mathbf{HeurBPP}, \mathbf{PSamp})$.

4. Если $(\mathbf{HeurBPP}, \mathbf{PSamp}) \subseteq (\mathbf{Heur}_{\frac{1}{n^c}}\mathbf{P}, \mathbf{PSamp})$, то $(\mathbf{HeurP}, \mathbf{PSamp}) = (\mathbf{HeurBPP}, \mathbf{PSamp})$.

Доказательство. Доказательство аналогично доказательствам теоремы 4.2, следствия 4.1, теоремы 4.3 и следствия 4.2. □

Отметим, что задача, полная в классе $(\mathbf{AvgBPP}, \mathbf{PSamp})$ и $(\mathbf{HeurBPP}, \mathbf{PSamp})$ одна и та же. В частности, это означает, что если $(\mathbf{AvgBPP}, \mathbf{PSamp}) \subseteq (\mathbf{HeurP}, \mathbf{PSamp})$, то $(C, R) \in (\mathbf{HeurP}, \mathbf{PSamp})$ и $(\mathbf{HeurP}, \mathbf{PSamp}) = (\mathbf{HeurBPP}, \mathbf{PSamp})$. Если $(\mathbf{AvgP}, \mathbf{PSamp}) = (\mathbf{AvgBPP}, \mathbf{PSamp})$, то поскольку $\mathbf{AvgP} \subseteq \mathbf{HeurP}$, получаем $(\mathbf{HeurP}, \mathbf{PSamp}) = (\mathbf{HeurBPP}, \mathbf{PSamp})$.

Список литературы

[Bar02] Boaz Barak. A probabilistic-time hierarchy theorem for “slightly non-uniform” algorithms. In *RANDOM '02: Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, pages 194–208, London, UK, 2002. Springer-Verlag.

[BDCGL92] Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. On the theory of average case complexity. *J. Comput. Syst. Sci.*, 44(2):193–219, 1992.

[BT06] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Foundation and Trends in Theoretical Computer Science*, 2(1):1–106, 2006.

[FS04] Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *FOCS*, pages 316–324, 2004.

[GHP06] Dima Grigoriev, Edward A. Hirsch, and K. Pervyshev. A complete public-key cryptosystem. Technical Report 06-046, Electronic Colloquium on Computational Complexity, 2006.

[HH86] Juris Hartmanis and Lane A. Hemachandra. Complexity classes without machines: On complete languages for up. In *ICALP '86: Proceedings of the 13th International Colloquium on Automata, Languages and Programming*, pages 123–135, London, UK, 1986. Springer-Verlag.

[HKN+05] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In *EUROCRYPT*, pages 96–113, 2005.

[Imp95] R. Impagliazzo. A personal view of average-case complexity. In *SCT '95: Proceedings of the 10th Annual Structure in Complexity Theory Conference (SCT'95)*, page 134, Washington, DC, USA, 1995. IEEE Computer Society.

[KV87] Marek Karpinski and Rutger Verbeek. *Randomness, provability, and the separation of Monte Carlo time and space*, pages 189–207. Springer-Verlag, London, UK, 1987.

[Mil01] Peter Bro Miltersen. *Handbook on Randomization*, volume II, chapter 19. Derandomizing Complexity Classes. Kluwer Academic Publishers, July 2001.

[Per07] Konstantin Pervyshev. On heuristic time hierarchies. In *IEEE Conference on Computational Complexity*, pages 347–358, 2007.