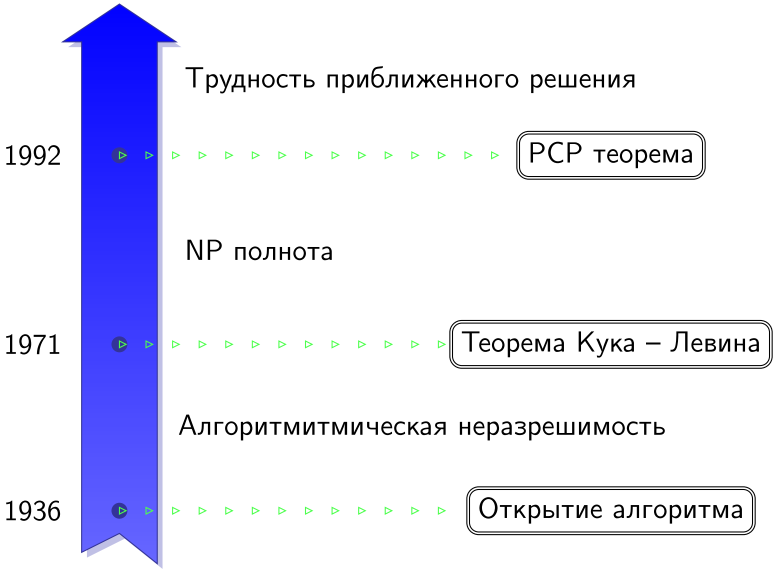
PCP теорема. Часть 1

М. Вялый

МФТИ, 12.03.2013

Трудность решения задач: краткая история вопроса



План

Алгоритмические задачи: примеры задач разной трудности

Диофантовы уравнения

Дано: многочлен $f(x)$ от нескольких переменных с целыми коэффициентами.
Выяснить: имеет ли уравнение $f(x) = 0$ целочисленные решения?

Линейные диофантовы уравнения

Дано: линейные многочлены $f_i(x)$ от нескольких переменных с целыми коэффициентами.
Выяснить: имеет ли система уравнений $f_i(x) = 0$ целочисленные решения?

Исходный вопрос теории алгоритмов: существует ли алгоритм решения некоторой алгоритмической задачи?

Теорема (Davis, Putnam, Robinson, Матиясевич, 1970)

Задача «Диофантовы уравнения» алгоритмически неразрешима.

Утверждение

Задача «Линейные диофантовы уравнения» алгоритмически разрешима.

Теория алгоритмов: примеры результатов

Исходный вопрос теории алгоритмов: существует ли алгоритм решения некоторой алгоритмической задачи?

Теорема (Davis, Putnam, Robinson, Матиясевич, 1970)

Задача «Диофантовы уравнения» алгоритмически неразрешима.

Утверждение

Задача «Линейные диофантовы уравнения» алгоритмически разрешима.

Теория алгоритмов: примеры результатов

Исходный вопрос теории алгоритмов: существует ли алгоритм решения некоторой алгоритмической задачи?

Теорема (Davis, Putnam, Robinson, Матиясевич, 1970)

Задача «Диофантовы уравнения» алгоритмически неразрешима.

Утверждение

Задача «Линейные диофантовы уравнения» алгоритмически разрешима.

Теория вычислительной сложности: примеры результатов и гипотез

Исходный вопрос: существует ли эффективный алгоритм решения алгоритмической задачи?

Утверждение

Для задачи «Линейные диофантовы уравнения» существует эффективный алгоритм решения.

Задача выполнимости КНФ

Дано: КНФ (конъюнкция дизъюнкций литералов).
Выяснить: существует ли выполняющий набор значений переменных?

Гипотеза

Для выполнимости КНФ не существует эффективного алгоритма решения.

Теория вычислительной сложности: примеры результатов и гипотез

Исходный вопрос: существует ли эффективный алгоритм решения алгоритмической задачи?

Утверждение

Для задачи «Линейные диофантовы уравнения» существует эффективный алгоритм решения.

Задача выполнимости КНФ

Дано: КНФ (конъюнкция дизъюнкций литералов).
Выяснить: существует ли выполняющий набор значений переменных?

Гипотеза

Для выполнимости КНФ не существует эффективного алгоритма решения.

Теория вычислительной сложности: примеры результатов и гипотез

Исходный вопрос: существует ли эффективный алгоритм решения алгоритмической задачи?

Утверждение

Для задачи «Линейные диофантовы уравнения» существует эффективный алгоритм решения.

Задача выполнимости КНФ

Дано: КНФ (конъюнкция дизъюнкций литералов).
Выяснить: существует ли выполняющий набор значений переменных?

Гипотеза

Для выполнимости КНФ не существует эффективного алгоритма решения.

Теория вычислительной сложности: примеры результатов и гипотез

Исходный вопрос: существует ли эффективный алгоритм решения алгоритмической задачи?

Утверждение

Для задачи «Линейные диофантовы уравнения» существует эффективный алгоритм решения.

Задача выполнимости КНФ

Дано: КНФ (конъюнкция дизъюнкций литералов).
Выяснить: существует ли выполняющий набор значений переменных?

Гипотеза

Для выполнимости КНФ не существует эффективного алгоритма решения.

Условные результаты

Задача выполнимости 3-КНФ

Дано: 3-КНФ: конъюнкция дизъюнкций (в точности) трех литералов.
Выяснить: существует ли выполняющий набор значений переменных?

Утверждение

Если существует эффективный алгоритм решения задачи выполнимости 3-КНФ, то существует эффективный алгоритм решения задачи выполнимости КНФ.

Условные результаты

Задача выполнимости 3-КНФ

Дано: 3-КНФ: конъюнкция дизъюнкций (в точности) трех литералов.
Выяснить: существует ли выполняющий набор значений переменных?

Утверждение

Если существует эффективный алгоритм решения задачи выполнимости 3-КНФ, то существует эффективный алгоритм решения задачи выполнимости КНФ.

Задачи с априорной информацией

Вычисление частично определенного предиката.
О входных данных заранее известна некоторая информация.
Интересует (эффективный) алгоритм решения, корректно работающий на таких входах.

Задача MAX-3SAT(a, b)

Дано: 3-КНФ $C = \bigwedge_{i=1}^{m} C_i$.

Известно:

- либо существует такой набор значений переменных, что $\geqslant bm$ дизъюнктов истинны на этом наборе,
- либо на любом наборе значений переменных истинны $< am$ дизъюнктов.

Выяснить: какой из вариантов имеет место.

Задачи с априорной информацией

Вычисление частично определенного предиката.
О входных данных заранее известна некоторая информация.
Интересует (эффективный) алгоритм решения, корректно работающий на таких входах.

Задача MAX-3SAT(a, b)

Дано: 3-КНФ $C = \bigwedge_{i=1}^{m} C_i$.
Известно:

- либо существует такой набор значений переменных, что $\geqslant bm$ дизъюнктов истинны на этом наборе,
- либо на любом наборе значений переменных истинны $< am$ дизъюнктов.

Выяснить: какой из вариантов имеет место.

Точная граница условной трудности

Теорема (Johnson, 1973)

Для задачи MAX-3SAT$(7/8, 1)$ существует эффективный алгоритм решения.

Теорема (Håstad, 2001)

Для любого $\varepsilon > 0$ из существования эффективного алгоритма решения задачи MAX-3SAT$(7/8 + \varepsilon, 1)$ следует существование эффективного алгоритма решения задачи выполнимости КНФ.

Одно из самых выдающихся достижений теории сложности. Доказательство очень непростое, основано на комбинации нескольких глубоких теорем. Одна из них — основная тема этого рассказа.

Точная граница условной трудности

Теорема (Johnson, 1973)

Для задачи MAX-3SAT$(7/8, 1)$ существует эффективный алгоритм решения.

Теорема (Håstad, 2001)

Для любого $\varepsilon > 0$ из существования эффективного алгоритма решения задачи MAX-3SAT$(7/8 + \varepsilon, 1)$ следует существование эффективного алгоритма решения задачи выполнимости КНФ.

Одно из самых выдающихся достижений теории сложности. Доказательство очень непростое, основано на комбинации нескольких глубоких теорем. Одна из них — основная тема этого рассказа.

Точная граница условной трудности

Теорема (Johnson, 1973)

Для задачи MAX-3SAT$(7/8, 1)$ существует эффективный алгоритм решения.

Теорема (Håstad, 2001)

Для любого $\varepsilon > 0$ из существования эффективного алгоритма решения задачи MAX-3SAT$(7/8 + \varepsilon, 1)$ следует существование эффективного алгоритма решения задачи выполнимости КНФ.

Одно из самых выдающихся достижений теории сложности. Доказательство очень непростое, основано на комбинации нескольких глубоких теорем. Одна из них — основная тема этого рассказа.

План

О моделях вычисления

Алгоритм — способ вычисления (частично определенной) функции $f\colon \Sigma^* \to \Sigma^*$, $|\Sigma| < \infty$. Без ограничения общности $\Sigma = \{0,1\}$.
Задачи разрешения: вычисление характеристической функции множества $\chi_L\colon \{0,1\}^* \to \{0,1\}$, $L \subseteq \{0,1\}^*$. Множество (или язык) состоит из тех входов алгоритмической задачи, для которых ответ на вопрос задачи положительный.

Несколько замечаний:

- Все «разумные» модели вычисления эквивалентны.
- Основная модель: многоленточная машина Тьюринга. Время работы: количество тактов.
- Модели с адресным доступом ко входу. МТ пишет на специальную адресную ленту двоичную запись числа k и по запросу получает значение k-го бита входа.

О моделях вычисления

Алгоритм — способ вычисления (частично определенной) функции $f\colon \Sigma^* \to \Sigma^*$, $|\Sigma| < \infty$. Без ограничения общности $\Sigma = \{0, 1\}$.
Задачи разрешения: вычисление характеристической функции множества $\chi_L\colon \{0,1\}^* \to \{0,1\}$, $L \subseteq \{0,1\}^*$. Множество (или язык) состоит из тех входов алгоритмической задачи, для которых ответ на вопрос задачи положительный.
Несколько замечаний:

- Все «разумные» модели вычисления эквивалентны.
- Основная модель: многоленточная машина Тьюринга. Время работы: количество тактов.
- Модели с адресным доступом ко входу. МТ пишет на специальную адресную ленту двоичную запись числа k и по запросу получает значение k-го бита входа.

О моделях вычисления

Алгоритм — способ вычисления (частично определенной) функции $f\colon \Sigma^* \to \Sigma^*$, $|\Sigma| < \infty$. Без ограничения общности $\Sigma = \{0, 1\}$.
Задачи разрешения: вычисление характеристической функции множества $\chi_L\colon \{0,1\}^* \to \{0,1\}$, $L \subseteq \{0,1\}^*$. Множество (или язык) состоит из тех входов алгоритмической задачи, для которых ответ на вопрос задачи положительный.
Несколько замечаний:

- Все «разумные» модели вычисления эквивалентны.
- Основная модель: многоленточная машина Тьюринга. Время работы: количество тактов.
- Модели с адресным доступом ко входу. МТ пишет на специальную адресную ленту двоичную запись числа k и по запросу получает значение k-го бита входа.

О моделях вычисления

Алгоритм — способ вычисления (частично определенной) функции $f\colon \Sigma^* \to \Sigma^*$, $|\Sigma| < \infty$. Без ограничения общности $\Sigma = \{0,1\}$.
Задачи разрешения: вычисление характеристической функции множества $\chi_L\colon \{0,1\}^* \to \{0,1\}$, $L \subseteq \{0,1\}^*$. Множество (или язык) состоит из тех входов алгоритмической задачи, для которых ответ на вопрос задачи положительный.
Несколько замечаний:

- Все «разумные» модели вычисления эквивалентны.
- Основная модель: многоленточная машина Тьюринга. Время работы: количество тактов.
- Модели с адресным доступом ко входу. МТ пишет на специальную адресную ленту двоичную запись числа k и по запросу получает значение k-го бита входа.

Класс $\mathrm{DTIME}(f(n))$

$L \in \mathrm{DTIME}(f(n))$ если и только если существует такой алгоритм разрешения L, который на входе длины n работает за время $O(f(n))$.

Класс P

$$\mathrm{P} = \bigcup_k \mathrm{DTIME}(n^k) \stackrel{\text{def}}{=} \mathrm{DTIME}(\mathrm{poly}(n)).$$

Определение эффективного алгоритма

Алгоритм эффективный, если он работает за время $\mathrm{poly}(n)$.

Ресурсы. Время

Класс $\mathrm{DTIME}(f(n))$

$L \in \mathrm{DTIME}(f(n))$ если и только если существует такой алгоритм разрешения L, который на входе длины n работает за время $O(f(n))$.

Класс P

$$\mathrm{P} = \bigcup_k \mathrm{DTIME}(n^k) \stackrel{\text{def}}{=} \mathrm{DTIME}(\mathrm{poly}(n)).$$

Определение эффективного алгоритма

Алгоритм эффективный, если он работает за время $\mathrm{poly}(n)$.

Ресурсы. Время

Класс $\mathrm{DTIME}(f(n))$

$L \in \mathrm{DTIME}(f(n))$ если и только если существует такой алгоритм разрешения L, который на входе длины n работает за время $O(f(n))$.

Класс P

$$\mathrm{P} = \bigcup_k \mathrm{DTIME}(n^k) \stackrel{\text{def}}{=} \mathrm{DTIME}(\mathrm{poly}(n)).$$

Определение эффективного алгоритма

Алгоритм эффективный, если он работает за время $\mathrm{poly}(n)$.

Ресурсы. Случайность

- Вероятностный алгоритм имеет доступ к генератору случайных битов. Стандартный генератор выдает значения битов равновероятно, и эти значения независимы для разных запросов.
- На результатах работы вероятностного алгоритма возникает вероятностное распределение.

Класс BPP

$L \in \mathrm{BPP}$, если существует такой вероятностный алгоритм $A\colon \{0,1\}^* \to \{0,1\}$, работающий за время $\mathrm{poly}(n)$, что:

- $x \in L \Rightarrow \mathbf{Pr}[A(x) = 1] > 2/3$;
- $x \notin L \Rightarrow \mathbf{Pr}[A(x) = 1] < 1/3$;

Одна из основных нерешенных проблем теории сложности

Равны ли классы P и BPP?

Ресурсы. Случайность

- Вероятностный алгоритм имеет доступ к генератору случайных битов. Стандартный генератор выдает значения битов равновероятно, и эти значения независимы для разных запросов.
- На результатах работы вероятностного алгоритма возникает вероятностное распределение.

Класс BPP (другой вариант эффективного алгоритма)

$L \in \mathrm{BPP}$, если существует такой вероятностный алгоритм $A\colon \{0,1\}^* \to \{0,1\}$, работающий за время $\mathrm{poly}(n)$, что:

- $x \in L \Rightarrow \mathbf{Pr}[A(x) = 1] > 2/3$;
- $x \notin L \Rightarrow \mathbf{Pr}[A(x) = 1] < 1/3$;

Одна из основных нерешенных проблем теории сложности

Равны ли классы P и BPP?

Ресурсы. Случайность

- Вероятностный алгоритм имеет доступ к генератору случайных битов. Стандартный генератор выдает значения битов равновероятно, и эти значения независимы для разных запросов.
- На результатах работы вероятностного алгоритма возникает вероятностное распределение.

Класс BPP (другой вариант эффективного алгоритма)

$L \in \mathrm{BPP}$, если существует такой вероятностный алгоритм $A\colon \{0,1\}^* \to \{0,1\}$, работающий за время $\mathrm{poly}(n)$, что:

- $x \in L \Rightarrow \mathbf{Pr}[A(x) = 1] > 2/3$;
- $x \notin L \Rightarrow \mathbf{Pr}[A(x) = 1] < 1/3$;

Одна из основных нерешенных проблем теории сложности

Равны ли классы P и BPP?

Ресурсы. Случайность

- Вероятностный алгоритм имеет доступ к генератору случайных битов. Стандартный генератор выдает значения битов равновероятно, и эти значения независимы для разных запросов.
- На результатах работы вероятностного алгоритма возникает вероятностное распределение.

Класс BPP (другой вариант эффективного алгоритма)

$L \in \mathrm{BPP}$, если существует такой вероятностный алгоритм $A\colon \{0,1\}^* \to \{0,1\}$, работающий за время $\mathrm{poly}(n)$, что:

- $x \in L \Rightarrow \mathbf{Pr}[A(x) = 1] > 2/3$;
- $x \notin L \Rightarrow \mathbf{Pr}[A(x) = 1] < 1/3$;

Одна из основных нерешенных проблем теории сложности

Равны ли классы P и BPP?

Ресурсы. Случайность

- Вероятностный алгоритм имеет доступ к генератору случайных битов. Стандартный генератор выдает значения битов равновероятно, и эти значения независимы для разных запросов.
- На результатах работы вероятностного алгоритма возникает вероятностное распределение.

Класс BPP (другой вариант эффективного алгоритма)

$L \in \mathrm{BPP}$, если существует такой вероятностный алгоритм $A\colon \{0,1\}^* \to \{0,1\}$, работающий за время $\mathrm{poly}(n)$, что:

- $x \in L \Rightarrow \mathbf{Pr}[A(x) = 1] > 2/3$;
- $x \notin L \Rightarrow \mathbf{Pr}[A(x) = 1] < 1/3$;

Одна из основных нерешенных проблем теории сложности

Равны ли классы P и BPP?

Ресурсы. Оракулы и советники

- Оракул вычисляет значение некоторой функции за один такт работы. (Отвечает правильно на вопрос.)
- Советник (prover) всё знает, но он предвзят. Он стремится к тому, чтобы ответ алгоритма (verifier'а) был «да».
- Протокол работы с Советником — последовательность битов. Если алгоритм детерминированный, Советник может заранее промоделировать протокол и выдать его алгоритму с самого начала работы.
- Поэтому стандартный вид алгоритма с доступом к Советнику: алгоритм с двумя входами x, y.
- x — это входные данные.
- y — последовательность ответов Советника (доказательство).

Ресурсы. Оракулы и советники

- Оракул вычисляет значение некоторой функции за один такт работы. (Отвечает правильно на вопрос.)
- Советник (prover) всё знает, но он предвзят. Он стремится к тому, чтобы ответ алгоритма (verifier'а) был «да».
- Протокол работы с Советником — последовательность битов. Если алгоритм детерминированный, Советник может заранее промоделировать протокол и выдать его алгоритму с самого начала работы.
- Поэтому стандартный вид алгоритма с доступом к Советнику: алгоритм с двумя входами x, y.
- x — это входные данные.
- y — последовательность ответов Советника (доказательство).

Ресурсы. Оракулы и советники

- Оракул вычисляет значение некоторой функции за один такт работы. (Отвечает правильно на вопрос.)
- Советник (prover) всё знает, но он предвзят. Он стремится к тому, чтобы ответ алгоритма (verifier'а) был «да».
- Протокол работы с Советником — последовательность битов. Если алгоритм детерминированный, Советник может заранее промоделировать протокол и выдать его алгоритму с самого начала работы.
- Поэтому стандартный вид алгоритма с доступом к Советнику: алгоритм с двумя входами x, y.
- x — это входные данные.
- y — последовательность ответов Советника (доказательство).

Ресурсы. Оракулы и советники

- Оракул вычисляет значение некоторой функции за один такт работы. (Отвечает правильно на вопрос.)
- Советник (prover) всё знает, но он предвзят. Он стремится к тому, чтобы ответ алгоритма (verifier'а) был «да».
- Протокол работы с Советником — последовательность битов. Если алгоритм детерминированный, Советник может заранее промоделировать протокол и выдать его алгоритму с самого начала работы.
- Поэтому стандартный вид алгоритма с доступом к Советнику: алгоритм с двумя входами x, y.
- x — это входные данные.
- y — последовательность ответов Советника (доказательство).

Ресурсы. Оракулы и советники

- Оракул вычисляет значение некоторой функции за один такт работы. (Отвечает правильно на вопрос.)
- Советник (prover) всё знает, но он предвзят. Он стремится к тому, чтобы ответ алгоритма (verifier'а) был «да».
- Протокол работы с Советником — последовательность битов. Если алгоритм детерминированный, Советник может заранее промоделировать протокол и выдать его алгоритму с самого начала работы.
- Поэтому стандартный вид алгоритма с доступом к Советнику: алгоритм с двумя входами x, y.
- x — это входные данные.
- y — последовательность ответов Советника (доказательство).

Ресурсы. Оракулы и советники

- Оракул вычисляет значение некоторой функции за один такт работы. (Отвечает правильно на вопрос.)
- Советник (prover) всё знает, но он предвзят. Он стремится к тому, чтобы ответ алгоритма (verifier'a) был «да».
- Протокол работы с Советником — последовательность битов. Если алгоритм детерминированный, Советник может заранее промоделировать протокол и выдать его алгоритму с самого начала работы.
- Поэтому стандартный вид алгоритма с доступом к Советнику: алгоритм с двумя входами x, y.
- x — это входные данные.
- y — последовательность ответов Советника (доказательство).

Класс NP

Класс NP

$L \in \mathrm{NP}$ если и только если существует такой алгоритм (многоленточная МТ) $V\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$, что

- на входе (x, y) алгоритм работает за время $\mathrm{poly}(|x|)$;
- $x \in L \Rightarrow \exists y : V(x, y) = 1$ (Советник может убедить алгоритм выдать ответ «да»);
- $x \notin L \Rightarrow \forall y : V(x, y) = 0$ (чтобы ни говорил Советник, алгоритм выдаст ответ «нет»).

Задача

Докажите, что если заменить многоленточную МТ на машину с произвольным (адресным) доступом к доказательству, класс NP не изменится.

Класс NP

Класс NP

$L \in \mathrm{NP}$ если и только если существует такой алгоритм (многоленточная МТ) $V\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$, что

- на входе (x, y) алгоритм работает за время $\mathrm{poly}(|x|)$;
- $x \in L \Rightarrow \exists y : V(x, y) = 1$ (Советник может убедить алгоритм выдать ответ «да»);
- $x \notin L \Rightarrow \forall y : V(x, y) = 0$ (чтобы ни говорил Советник, алгоритм выдаст ответ «нет»).

Задача

Докажите, что если заменить многоленточную МТ на машину с произвольным (адресным) доступом к доказательству, класс NP не изменится.

Классы PCP, неформальное введение

Probabilistically checkable proofs (вероятностно проверяемые доказательства).

- Доказательство может быть очень длинным. Хочется проверить его правильность, не изучая досконально все детали.
- Чтобы результат проверки зависел от всего доказательства, можно выбирать случайно места для проверки.
- Чтобы от таких случайных проверок был прок, нужен специальный формат записи доказательства.

Вопрос

Есть ли такой формат записи доказательства, который позволяет после чтения лишь небольшой части текста доказательства со значительной уверенностью судить о корректности доказательства?

Классы PCP, неформальное введение

Probabilistically checkable proofs (вероятностно проверяемые доказательства).

- Доказательство может быть очень длинным. Хочется проверить его правильность, не изучая досконально все детали.
- Чтобы результат проверки зависел от всего доказательства, можно выбирать случайно места для проверки.
- Чтобы от таких случайных проверок был прок, нужен специальный формат записи доказательства.

Вопрос

Есть ли такой формат записи доказательства, который позволяет после чтения лишь небольшой части текста доказательства со значительной уверенностью судить о корректности доказательства?

Классы PCP, неформальное введение

Probabilistically checkable proofs (вероятностно проверяемые доказательства).

- Доказательство может быть очень длинным. Хочется проверить его правильность, не изучая досконально все детали.
- Чтобы результат проверки зависел от всего доказательства, можно выбирать случайно места для проверки.
- Чтобы от таких случайных проверок был прок, нужен специальный формат записи доказательства.

Вопрос

Есть ли такой формат записи доказательства, который позволяет после чтения лишь небольшой части текста доказательства со значительной уверенностью судить о корректности доказательства?

Классы PCP, неформальное введение

Probabilistically checkable proofs (вероятностно проверяемые доказательства).

- Доказательство может быть очень длинным. Хочется проверить его правильность, не изучая досконально все детали.
- Чтобы результат проверки зависел от всего доказательства, можно выбирать случайно места для проверки.
- Чтобы от таких случайных проверок был прок, нужен специальный формат записи доказательства.

Вопрос

Есть ли такой формат записи доказательства, который позволяет после чтения лишь небольшой части текста доказательства со значительной уверенностью судить о корректности доказательства?

Классы $\mathrm{PCP}(r(n), q(n))$, адаптивное определение

Рассматриваем (вероятностные) алгоритмы с адресным доступом к доказательству, которые на входе длины n выполняют запросы к генератору случайных битов (не более $O(r(n))$ штук) и к доказательству (не более $O(q(n))$ штук). Вход читается без ограничений.

Класс $\mathrm{PCP}(r(n), q(n))$

$L \in \mathrm{PCP}(r(n), q(n))$ если и только если существует такой алгоритм указанного вида $V\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$, что

- на входе (x, y) алгоритм работает за время $\mathrm{poly}(n)$;
- $x \in L \Rightarrow \exists y : \mathbf{Pr}[V(x,y) = 1] = 1$ (существует корректное доказательство);
- $x \notin L \Rightarrow \forall y : \mathbf{Pr}[V(x,y) = 1] \leqslant 1/2$ (любое доказательство отвергается с достаточно большой вероятностью).

Классы $\mathrm{PCP}(r(n), q(n))$, адаптивное определение

Рассматриваем (вероятностные) алгоритмы с адресным доступом к доказательству, которые на входе длины n выполняют запросы к генератору случайных битов (не более $O(r(n))$ штук) и к доказательству (не более $O(q(n))$ штук). Вход читается без ограничений.

Класс $\mathrm{PCP}(r(n), q(n))$

$L \in \mathrm{PCP}(r(n), q(n))$ если и только если существует такой алгоритм указанного вида $V\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$, что

- на входе (x, y) алгоритм работает за время $\mathrm{poly}(n)$;
- $x \in L \Rightarrow \exists y : \mathbf{Pr}[V(x,y) = 1] = 1$ (существует корректное доказательство);
- $x \notin L \Rightarrow \forall y : \mathbf{Pr}[V(x,y) = 1] \leqslant 1/2$ (любое доказательство отвергается с достаточно большой вероятностью).

Классы $\mathrm{PCP}(r(n), q(n))$, адаптивное определение

Рассматриваем (вероятностные) алгоритмы с адресным доступом к доказательству, которые на входе длины n выполняют запросы к генератору случайных битов (не более $O(r(n))$ штук) и к доказательству (не более $O(q(n))$ штук). Вход читается без ограничений.

Класс $\mathrm{PCP}(r(n), q(n))$

$L \in \mathrm{PCP}(r(n), q(n))$ если и только если существует такой алгоритм указанного вида $V\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$, что

- на входе (x, y) алгоритм работает за время $\mathrm{poly}(n)$;
- $x \in L \Rightarrow \exists y : \mathbf{Pr}[V(x, y) = 1] = 1$ (существует корректное доказательство);
- $x \notin L \Rightarrow \forall y : \mathbf{Pr}[V(x, y) = 1] \leqslant 1/2$ (любое доказательство отвергается с достаточно большой вероятностью).

PCP теорема

Теорема (Arora, Lund, Motwani, Sudan, Szegedy, 1992)

$\mathrm{PCP}(\log n, 1) = \mathrm{NP}$.

Легкая часть $\mathrm{PCP}(\log n, 1) \subseteq \mathrm{NP}$

- Разных наборов значений случайных битов $\mathrm{poly}(n)$. Прочитанных битов доказательства тоже $\mathrm{poly}(n)$. NP-алгоритм угадывает все их значения и имитирует работу PCP-алгоритма.
- NP-алгоритм выдает ответ «да», если при всех возможных значениях случайных битов PCP-алгоритм выдает ответ «да». В противном случае ответ «нет».
- Если существует корректное доказательство, то ответ у такого NP-алгоритма будет «да».
- Если корректного доказательства не существует, то NP-алгоритм всегда выдает ответ «нет».

PCP теорема

Теорема (Arora, Lund, Motwani, Sudan, Szegedy, 1992)

$\mathrm{PCP}(\log n, 1) = \mathrm{NP}$.

Легкая часть $\mathrm{PCP}(\log n, 1) \subseteq \mathrm{NP}$

- Разных наборов значений случайных битов $\mathrm{poly}(n)$. Прочитанных битов доказательства тоже $\mathrm{poly}(n)$. NP-алгоритм угадывает все их значения и имитирует работу PCP-алгоритма.
- NP-алгоритм выдает ответ «да», если при всех возможных значениях случайных битов PCP-алгоритм выдает ответ «да». В противном случае ответ «нет».
- Если существует корректное доказательство, то ответ у такого NP-алгоритма будет «да».
- Если корректного доказательства не существует, то NP-алгоритм всегда выдает ответ «нет».

PCP теорема

Теорема (Arora, Lund, Motwani, Sudan, Szegedy, 1992)

$\mathrm{PCP}(\log n, 1) = \mathrm{NP}$.

Легкая часть $\mathrm{PCP}(\log n, 1) \subseteq \mathrm{NP}$

- Разных наборов значений случайных битов $\mathrm{poly}(n)$. Прочитанных битов доказательства тоже $\mathrm{poly}(n)$. NP-алгоритм угадывает все их значения и имитирует работу PCP-алгоритма.
- NP-алгоритм выдает ответ «да», если при всех возможных значениях случайных битов PCP-алгоритм выдает ответ «да». В противном случае ответ «нет».
- Если существует корректное доказательство, то ответ у такого NP-алгоритма будет «да».
- Если корректного доказательства не существует, то NP-алгоритм всегда выдает ответ «нет».

Теорема (Arora, Lund, Motwani, Sudan, Szegedy, 1992)

$\mathrm{PCP}(\log n, 1) = \mathrm{NP}$.

Легкая часть $\mathrm{PCP}(\log n, 1) \subseteq \mathrm{NP}$

- Разных наборов значений случайных битов $\mathrm{poly}(n)$. Прочитанных битов доказательства тоже $\mathrm{poly}(n)$. NP-алгоритм угадывает все их значения и имитирует работу PCP-алгоритма.
- NP-алгоритм выдает ответ «да», если при всех возможных значениях случайных битов PCP-алгоритм выдает ответ «да». В противном случае ответ «нет».
- Если существует корректное доказательство, то ответ у такого NP-алгоритма будет «да».
- Если корректного доказательства не существует, то NP-алгоритм всегда выдает ответ «нет».

PCP теорема

Теорема (Arora, Lund, Motwani, Sudan, Szegedy, 1992)

$\mathrm{PCP}(\log n, 1) = \mathrm{NP}$.

Легкая часть $\mathrm{PCP}(\log n, 1) \subseteq \mathrm{NP}$

- Разных наборов значений случайных битов $\mathrm{poly}(n)$. Прочитанных битов доказательства тоже $\mathrm{poly}(n)$. NP-алгоритм угадывает все их значения и имитирует работу PCP-алгоритма.
- NP-алгоритм выдает ответ «да», если при всех возможных значениях случайных битов PCP-алгоритм выдает ответ «да». В противном случае ответ «нет».
- Если существует корректное доказательство, то ответ у такого NP-алгоритма будет «да».
- Если корректного доказательства не существует, то NP-алгоритм всегда выдает ответ «нет».

PCP теорема

Теорема (Arora, Lund, Motwani, Sudan, Szegedy, 1992)

$\mathrm{PCP}(\log n, 1) = \mathrm{NP}$.

Легкая часть $\mathrm{PCP}(\log n, 1) \subseteq \mathrm{NP}$

- Разных наборов значений случайных битов $\mathrm{poly}(n)$. Прочитанных битов доказательства тоже $\mathrm{poly}(n)$. NP-алгоритм угадывает все их значения и имитирует работу PCP-алгоритма.
- NP-алгоритм выдает ответ «да», если при всех возможных значениях случайных битов PCP-алгоритм выдает ответ «да». В противном случае ответ «нет».
- Если существует корректное доказательство, то ответ у такого NP-алгоритма будет «да».
- Если корректного доказательства не существует, то NP-алгоритм всегда выдает ответ «нет».

Неадаптивные проверки

- Алгоритм составляет список из $O(q(n))$ адресов в доказательстве, используя $O(r(n))$ случайных битов.
- Алгоритм получает список значений битов в доказательстве в соответствии с указанными адресами.
- Алгоритм (детерминированно) принимает решение, изучая полученный список.

Усиленная PCP теорема

Для любого языка $L \in \mathrm{NP}$ существует неадаптивный $\mathrm{PCP}(\log n, 1)$-алгоритм проверки.

Неадаптивные проверки

- Алгоритм составляет список из $O(q(n))$ адресов в доказательстве, используя $O(r(n))$ случайных битов.
- Алгоритм получает список значений битов в доказательстве в соответствии с указанными адресами.
- Алгоритм (детерминированно) принимает решение, изучая полученный список.

Усиленная PCP теорема

Для любого языка $L \in \mathrm{NP}$ существует неадаптивный $\mathrm{PCP}(\log n, 1)$-алгоритм проверки.

Неадаптивные проверки

- Алгоритм составляет список из $O(q(n))$ адресов в доказательстве, используя $O(r(n))$ случайных битов.
- Алгоритм получает список значений битов в доказательстве в соответствии с указанными адресами.
- Алгоритм (детерминированно) принимает решение, изучая полученный список.

Усиленная PCP теорема

Для любого языка $L \in \mathrm{NP}$ существует неадаптивный $\mathrm{PCP}(\log n, 1)$-алгоритм проверки.

Неадаптивные проверки

- Алгоритм составляет список из $O(q(n))$ адресов в доказательстве, используя $O(r(n))$ случайных битов.
- Алгоритм получает список значений битов в доказательстве в соответствии с указанными адресами.
- Алгоритм (детерминированно) принимает решение, изучая полученный список.

Усиленная PCP теорема

Для любого языка $L \in \mathrm{NP}$ существует неадаптивный $\mathrm{PCP}(\log n, 1)$-алгоритм проверки.

План

Сводимости и сравнение трудности задач

Задачи отождествляем с языками.

Определение полиномиальной сводимости

$A \leqslant_p B$ если существует такое отображение $f\colon \Sigma^* \to \Sigma^*$, что

- $x \in A$ равносильно $f(x) \in B$;
- $f \in \mathrm{FP}$ (т. е. существует алгоритм, вычисляющий f за полиномиальное от длины входа время).

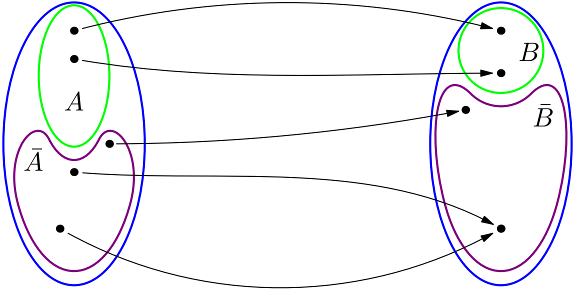
Сводимости и сравнение трудности задач

Задачи отождествляем с языками.

Определение полиномиальной сводимости

$A \leqslant_p B$ если существует такое отображение $f\colon \Sigma^* \to \Sigma^*$, что

- $x \in A$ равносильно $f(x) \in B$;
- $f \in \mathrm{FP}$ (т. е. существует алгоритм, вычисляющий f за полиномиальное от длины входа время).



Полные задачи

Определение полной в классе $\mathcal{C}$ задачи

A полна в классе $\mathcal{C}$, если $A \in \mathcal{C}$ и для любой $X \in \mathcal{C}$ выполняется
$$X \leqslant_p A.$$

Замечание

Полиномиальная сводимость и понятие полной задачи естественно обобщаются на задачи с априорной информацией.
Задаче с априорной информацией сопоставляется пара языков (L_0, L_1), $L_0 \cap L_1 = \varnothing$.
Определение. $(A_0, A_1) \leqslant_p (B_0, B_1)$ если существует такое отображение $f : \Sigma^* \to \Sigma^*$, что

- $x \in A_i$ равносильно $f(x) \in B_i$;
- $f \in \mathrm{FP}$ (т. е. существует алгоритм, вычисляющий f за полиномиальное от длины входа время).

Полные задачи

Определение полной в классе $\mathcal{C}$ задачи

A полна в классе $\mathcal{C}$, если $A \in \mathcal{C}$ и для любой $X \in \mathcal{C}$ выполняется

$$X \leqslant_p A.$$

Замечание

Полиномиальная сводимость и понятие полной задачи естественно обобщаются на задачи с априорной информацией.
Задаче с априорной информацией сопоставляется пара языков (L_0, L_1), $L_0 \cap L_1 = \varnothing$.
Определение. $(A_0, A_1) \leqslant_p (B_0, B_1)$ если существует такое отображение $f\colon \Sigma^* \to \Sigma^*$, что

- $x \in A_i$ равносильно $f(x) \in B_i$;
- $f \in \mathrm{FP}$ (т. е. существует алгоритм, вычисляющий f за полиномиальное от длины входа время).

О доказательстве NP-полноты задач

Нужен хотя бы один пример NP-полной задачи.

Теорема Кука – Левина

Выполнимость КНФ полна в классе NP.

Расширение списка с помощью сводимостей:

- Полиномиальная сводимость задает предпорядок (рефлексивное, транзитивное отношение).
- Для доказательства полноты задачи $B \in \mathrm{NP}$ достаточно построить полиномиальную сводимость $A \leqslant_p B$ какой-нибудь NP-полной задачи A к задаче B.
- Таким способом легко расширять список NP-полных задач. (Их многие тысячи!)

О доказательстве NP-полноты задач

Нужен хотя бы один пример NP-полной задачи.

Теорема Кука – Левина

Выполнимость КНФ полна в классе NP.

Расширение списка с помощью сводимостей:

- Полиномиальная сводимость задает предпорядок (рефлексивное, транзитивное отношение).
- Для доказательства полноты задачи $B \in \mathrm{NP}$ достаточно построить полиномиальную сводимость $A \leqslant_p B$ какой-нибудь NP-полной задачи A к задаче B.
- Таким способом легко расширять список NP-полных задач. (Их многие тысячи!)

О доказательстве NP-полноты задач

Нужен хотя бы один пример NP-полной задачи.

Теорема Кука – Левина

Выполнимость КНФ полна в классе NP.

Расширение списка с помощью сводимостей:

- Полиномиальная сводимость задает предпорядок (рефлексивное, транзитивное отношение).
- Для доказательства полноты задачи $B \in \mathrm{NP}$ достаточно построить полиномиальную сводимость $A \leqslant_p B$ какой-нибудь NP-полной задачи A к задаче B.
- Таким способом легко расширять список NP-полных задач. (Их многие тысячи!)

О доказательстве NP-полноты задач

Нужен хотя бы один пример NP-полной задачи.

Теорема Кука – Левина

Выполнимость КНФ полна в классе NP.

Расширение списка с помощью сводимостей:

- Полиномиальная сводимость задает предпорядок (рефлексивное, транзитивное отношение).
- Для доказательства полноты задачи $B \in \mathrm{NP}$ достаточно построить полиномиальную сводимость $A \leqslant_p B$ какой-нибудь NP-полной задачи A к задаче B.
- Таким способом легко расширять список NP-полных задач. (Их многие тысячи!)

О доказательстве NP-полноты задач

Нужен хотя бы один пример NP-полной задачи.

Теорема Кука – Левина

Выполнимость КНФ полна в классе NP.

Расширение списка с помощью сводимостей:

- Полиномиальная сводимость задает предпорядок (рефлексивное, транзитивное отношение).
- Для доказательства полноты задачи $B \in \mathrm{NP}$ достаточно построить полиномиальную сводимость $A \leqslant_p B$ какой-нибудь NP-полной задачи A к задаче B.
- Таким способом легко расширять список NP-полных задач. (Их многие тысячи!)

О доказательстве NP-полноты задач

Нужен хотя бы один пример NP-полной задачи.

Теорема Кука – Левина

Выполнимость КНФ полна в классе NP.

Расширение списка с помощью сводимостей:

- Полиномиальная сводимость задает предпорядок (рефлексивное, транзитивное отношение).
- Для доказательства полноты задачи $B \in \mathrm{NP}$ достаточно построить полиномиальную сводимость $A \leqslant_p B$ какой-нибудь NP-полной задачи A к задаче B.
- Таким способом легко расширять список NP-полных задач. (Их многие тысячи!)

План

Задачи оптимизации и задачи разрешения

Задача максимизации MAX-($\mathcal{M}$,$\mathcal{F}$)

Дано: множество $M \in \mathcal{M}$ и функция $f\colon M \to \mathbb{R}$, $f \in \mathcal{F}$.
Найти: $\max_{x \in M} f(x)$.

Соответствующая задача разрешения MAX-($\mathcal{M}$,$\mathcal{F}$)

Дано: рациональное число r, множество $M \in \mathcal{M}$ и функция $f\colon M \to \mathbb{R}$, $f \in \mathcal{F}$.

Выяснить: существует ли такое $x \in M$, что $f(x) \geqslant r$.

Соответствующая задача разрешения MAX-($\mathcal{M}$,$\mathcal{F}$)

Дано: рациональное число r, множество $M \in \mathcal{M}$ и функция $f\colon M \to \mathbb{R}$, $f \in \mathcal{F}$.

Выяснить: существует ли такое $x \in M$, что $f(x) \geqslant r$.

В дальнейшем мы рассматриваем только такие задачи MAX-($\mathcal{M}$,$\mathcal{F}$), которые принадлежат классу NP.

Задачи оптимизации и задачи разрешения

Соответствующая задача разрешения MAX-($\mathcal{M}$,$\mathcal{F}$)

Дано: рациональное число r, множество $M \in \mathcal{M}$ и функция $f : M \to \mathbb{R}$, $f \in \mathcal{F}$.

Выяснить: существует ли такое $x \in M$, что $f(x) \geqslant r$.

Замечание

Если (задача разрешения) MAX-($\mathcal{M}$,$\mathcal{F}$) $\in \mathrm{P}$ и известны априорные границы функционала

$$-\exp(\mathrm{poly}(n)) < \max_{x \in M} f(x) < \exp(\mathrm{poly}(n)),$$

то за время $\mathrm{poly}(n, \log(1/\varepsilon))$ можно вычислить $\max_{x \in M} f(x)$ с точностью ε (двоичный поиск).

Примеры

Задача MAX-3SAT

Дано: 3-КНФ $C = \bigwedge_{i=1}^{m} C_i$.
Найти: максимальное количество дизъюнктов, обращающихся в 1 на некотором наборе значений переменной.

Задача MAX-IND

Дано: Граф $G(V, E)$.
Найти: размер максимального независимого множества в G.

Задача MAX-CUT

Дано: Граф $G(V, E)$ и весовая функция на ребрах $w \colon E \to \mathbb{Q}_+$.
Найти: максимальный разрез

$$\max_{S \subseteq V} \sum_{\substack{e=(u,v)\in E \\ u\in S,\ v\in \bar{S}}} w(e).$$

Примеры

Задача MAX-3SAT

Дано: 3-КНФ $C = \bigwedge_{i=1}^{m} C_i$.
Найти: максимальное количество дизъюнктов, обращающихся в 1 на некотором наборе значений переменной.

Задача MAX-IND

Дано: Граф $G(V, E)$.
Найти: размер максимального независимого множества в G.

Задача MAX-CUT

Дано: Граф $G(V, E)$ и весовая функция на ребрах $w : E \to \mathbb{Q}_+$.
Найти: максимальный разрез

$$\max_{S \subseteq V} \sum_{\substack{e=(u,v)\in E\\ u\in S,\ v\in \bar{S}}} w(e).$$

Примеры

Задача MAX-3SAT

Дано: 3-КНФ $C = \bigwedge_{i=1}^{m} C_i$.
Найти: максимальное количество дизъюнктов, обращающихся в 1 на некотором наборе значений переменной.

Задача MAX-IND

Дано: Граф $G(V, E)$.
Найти: размер максимального независимого множества в G.

Задача MAX-CUT

Дано: Граф $G(V, E)$ и весовая функция на ребрах $w\colon E \to \mathbb{Q}_+$.
Найти: максимальный разрез

$$\max_{S \subseteq V} \sum_{\substack{e=(u,v)\in E\\ u\in S,\ v\in \bar{S}}} w(e).$$

Примеры

Задача MAX-3SAT (NP полная)

Дано: 3-КНФ $C = \bigwedge_{i=1}^{m} C_i$.
Найти: максимальное количество дизъюнктов, обращающихся в 1 на некотором наборе значений переменной.

Задача MAX-IND (NP полная)

Дано: Граф $G(V, E)$.
Найти: размер максимального независимого множества в G.

Задача MAX-CUT (NP полная)

Дано: Граф $G(V, E)$ и весовая функция на ребрах $w\colon E \to \mathbb{Q}_+$.
Найти: максимальный разрез

$$\max_{S\subseteq V} \sum_{\substack{e=(u,v)\in E\\ u\in S,\ v\in \bar{S}}} w(e).$$

Приближенные алгоритмы

Точность приближенного алгоритма

Обозначим через $I = (M, f)$ вход задачи MAX-($\mathcal{M}$,$\mathcal{F}$) с неотрицательными целевыми функциями. Алгоритм, вычисляющий функцию $A\colon I \to \mathbb{Q}$, имеет мультипликативную погрешность k, если

$$k \max_{x\in M} f(x) \leqslant A(I) \leqslant \max_{x\in M} f(x) \quad \text{для всех } I.$$

Точность приближения полиномиальными алгоритмами

MAX-3SAT	$7/8 = 0.875$	Johnson (1973)
MAX-CUT	$\min\limits_{0<\theta<\pi} \dfrac{\theta/\pi}{(1-\cos\theta)/2} \approx 0.88$	Goemans, Williamson (1995)
MAX-IND	$O\left(\dfrac{\log^3 n}{n(\log\log n)^2}\right)$	Feige (2004)

Приближенные алгоритмы

Точность приближенного алгоритма

Обозначим через $I = (M, f)$ вход задачи MAX-($\mathcal{M}$,$\mathcal{F}$) с неотрицательными целевыми функциями. Алгоритм, вычисляющий функцию $A\colon I \to \mathbb{Q}$, имеет мультипликативную погрешность k, если

$$k \max_{x\in M} f(x) \leqslant A(I) \leqslant \max_{x\in M} f(x) \quad \text{для всех } I.$$

Точность приближения полиномиальными алгоритмами

MAX-3SAT	$7/8 = 0.875$	Johnson (1973)
MAX-CUT	$\min\limits_{0<\theta<\pi} \dfrac{\theta/\pi}{(1-\cos\theta)/2} \approx 0.88$	Goemans, Williamson (1995)
MAX-IND	$O\left(\dfrac{\log^3 n}{n(\log\log n)^2}\right)$	Feige (2004)

Приближенные алгоритмы

SDP-релаксация: сведение комбинаторной задачи к задаче выпуклой оптимизации

Точность приближения полиномиальными алгоритмами

MAX-3SAT	$7/8 = 0.875$	Johnson (1973)
MAX-CUT	$\min\limits_{0<\theta<\pi} \dfrac{\theta/\pi}{(1-\cos\theta)/2} \approx 0.88$	Goemans, Williamson (1995)
MAX-IND	$O\left(\dfrac{\log^3 n}{n(\log\log n)^2}\right)$	Feige (2004)

Трудность приближения: примеры результатов

Результаты о трудности приближения основаны на PCP теореме и ее обобщениях.

В предположении $\mathrm{P} \neq \mathrm{NP}$ не существует полиномиального алгоритма с погрешностью k

Задача	k	
MAX-3SAT	$7/8+\varepsilon$ для любого $\varepsilon > 0$	Håstad (2001)
MAX-CUT	$16/17 \approx 0.94$	Håstad (2001)
MAX-IND	n^{-c} для какого-то $c > 0$	Arora, Lund, Motwani, Sudan, Szegedy (1998)

И в обратную сторону: доказать PCP теорему можно, доказав NP-трудность приближенного решения некоторой задачи.

Трудность приближения: примеры результатов

Результаты о трудности приближения основаны на PCP теореме и ее обобщениях.

В предположении $\mathrm{P} \neq \mathrm{NP}$ не существует полиномиального алгоритма с погрешностью k

Задача	k	
MAX-3SAT	$7/8 + \varepsilon$ для любого $\varepsilon > 0$	Håstad (2001)
MAX-CUT	$16/17 \approx 0.94$	Håstad (2001)
MAX-IND	n^{-c} для какого-то $c > 0$	Arora, Lund, Motwani, Sudan, Szegedy (1998)

И в обратную сторону: доказать PCP теорему можно, доказав NP-трудность приближенного решения некоторой задачи.

Трудность приближения: примеры результатов

Результаты о трудности приближения основаны на PCP теореме и ее обобщениях.

В предположении $\mathrm{P} \neq \mathrm{NP}$ не существует полиномиального алгоритма с погрешностью k

Задача	k	
MAX-3SAT	$7/8 + \varepsilon$ для любого $\varepsilon > 0$	Håstad (2001)
MAX-CUT	$16/17 \approx 0.94$	Håstad (2001)
MAX-IND	n^{-c} для какого-то $c > 0$	Arora, Lund, Motwani, Sudan, Szegedy (1998)

И в обратную сторону: доказать PCP теорему можно, доказав NP-трудность приближенного решения некоторой задачи.

План

Задача k-выполнимости: вход и присваивания

Гиперграф ограничений $G(V, E, \Sigma, c)$

- V — вершины графа (переменные)
- E — мультимножество ребер, каждое ребро $e \in V^k$
- Σ — алфавит (конечное множество)
- c — ограничения. Для каждого ребра $e \in E$ указано $c_e \subseteq \Sigma^k$.

Присваивания, выполнение ограничений, потери

- Присваивание: $\sigma : V \to \Sigma$
- Ограничение c_e, $e = (v_1, \dots, v_k)$, выполнено, если
$$(\sigma(v_1), \dots, \sigma(v_k)) \in c_e \subseteq \Sigma^k$$
- Потери на присваивании $\mathrm{UNSAT}_\sigma(G)$ — доля ограничений, не выполненных на σ
- Потери в задаче $\mathrm{UNSAT}(G) = \min_\sigma \mathrm{UNSAT}_\sigma(G)$

Задача k-выполнимости: вход и присваивания

Гиперграф ограничений $G(V, E, \Sigma, c)$

- V — вершины графа (переменные)
- E — мультимножество ребер, каждое ребро $e \in V^k$
- Σ — алфавит (конечное множество)
- c — ограничения. Для каждого ребра $e \in E$ указано $c_e \subseteq \Sigma^k$.

Присваивания, выполнение ограничений, потери

- Присваивание: $\sigma\colon V \to \Sigma$
- Ограничение c_e, $e = (v_1, \dots, v_k)$, выполнено, если
$$(\sigma(v_1), \dots, \sigma(v_k)) \in c_e \subseteq \Sigma^k$$
- Потери на присваивании $\mathrm{UNSAT}_\sigma(G)$ — доля ограничений, не выполненных на σ
- Потери в задаче $\mathrm{UNSAT}(G) = \min\limits_{\sigma} \mathrm{UNSAT}_\sigma(G)$

Задача k-выполнимости: формулировки

Задача MAX-kCSP$_q$

Дано: гиперграф ограничений $G(V, E, \Sigma, c)$, $E \subseteq V^k$, $|\Sigma| = q$.
Найти: $\text{UNSAT}(G)$.

Задача MAX-kCSP$_q(a, b)$ с «зазором»

Дано: гиперграф ограничений $G(V, E, \Sigma, c)$, $E \subseteq V^k$, $|\Sigma| = q$.
Известно:

- либо $\text{UNSAT}(G) \leqslant 1 - b$,
- либо $\text{UNSAT}(G) > 1 - a$.

Выяснить: какой из вариантов имеет место.

Задача k-выполнимости: формулировки

Задача MAX-kCSP$_q$

Дано: гиперграф ограничений $G(V, E, \Sigma, c)$, $E \subseteq V^k$, $|\Sigma| = q$.
Найти: $\textsc{unsat}(G)$.

Задача MAX-kCSP$_q(a, b)$ с «зазором»

Дано: гиперграф ограничений $G(V, E, \Sigma, c)$, $E \subseteq V^k$, $|\Sigma| = q$.
Известно:

- либо $\textsc{unsat}(G) \leqslant 1 - b$,
- либо $\textsc{unsat}(G) > 1 - a$.

Выяснить: какой из вариантов имеет место.

Примеры задач выполнимости

MAX-3SAT (арность 3, алфавит 2)

Ребра гиперграфа — тройки переменных, входящих в дизъюнкты.
Ограничения: $\ell_1 \vee \ell_2 \vee \ell_3 = 1$.

3-раскраска (арность 2, алфавит 3)

Ребра графа — ребра графа.
Ограничения: цвета концов ребра различны (одинаковое ограничение для всех ребер).

Примеры задач выполнимости

MAX-3SAT (арность 3, алфавит 2)

Ребра гиперграфа — тройки переменных, входящих в дизъюнкты.
Ограничения: $\ell_1 \vee \ell_2 \vee \ell_3 = 1$.

3-раскраска (арность 2, алфавит 3)

Ребра графа — ребра графа.
Ограничения: цвета концов ребра различны (одинаковое ограничение для всех ребер).

Важное наблюдение

Лемма

MAX-kCSP$_q(1-\alpha, 1) \in \mathrm{PCP}(\log n, 1)$. Здесь $\alpha = O(1)$.
Проверки неадаптивные.

Алгоритм проверки (ожидает доказательство в виде списка значений переменных в присваивании):

1. выбираем $C = \lceil -\log_2(1-\alpha) \rceil$ ограничений случайно и равновероятно;
2. запрашиваем значения соответствующих переменных, всего $O(kC \log q) = O(1)$ битов запрошено;
3. если все ограничения выполнены, то ответ «да», иначе «нет».

Если доказательство корректное, то ответ положительный всегда.
Если $\mathrm{UNSAT}(G) > \alpha$, то выполнено не более $1-\alpha$ доли ограничений на любом доказательстве; вероятность положительного ответа не превосходит

$$(1-\alpha)^C < \frac{1}{2}.$$

Важное наблюдение

Лемма

MAX-kCSP$_q(1-\alpha, 1) \in \mathrm{PCP}(\log n, 1)$. Здесь $\alpha = O(1)$. Проверки неадаптивные.

Алгоритм проверки (ожидает доказательство в виде списка значений переменных в присваивании):

1. выбираем $C = \lceil -\log_2(1-\alpha)\rceil$ ограничений случайно и равновероятно;
2. запрашиваем значения соответствующих переменных, всего $O(kC\log q) = O(1)$ битов запрошено;
3. если все ограничения выполнены, то ответ «да», иначе «нет».

Если доказательство корректное, то ответ положительный всегда. Если $\mathrm{UNSAT}(G) > \alpha$, то выполнено не более $1-\alpha$ доли ограничений на любом доказательстве; вероятность положительного ответа не превосходит

$$(1-\alpha)^C < \frac{1}{2}.$$

Важное наблюдение

Лемма

MAX-kCSP$_q(1-\alpha, 1) \in \mathrm{PCP}(\log n, 1)$. Здесь $\alpha = O(1)$.
Проверки неадаптивные.

Алгоритм проверки (ожидает доказательство в виде списка значений переменных в присваивании):

1. выбираем $C = \lceil -\log_2(1-\alpha) \rceil$ ограничений случайно и равновероятно;
2. запрашиваем значения соответствующих переменных, всего $O(kC \log q) = O(1)$ битов запрошено;
3. если все ограничения выполнены, то ответ «да», иначе «нет».

Если доказательство корректное, то ответ положительный всегда.

Если $\mathrm{UNSAT}(G) > \alpha$, то выполнено не более $1-\alpha$ доли ограничений на любом доказательстве; вероятность положительного ответа не превосходит

$$(1-\alpha)^C < \frac{1}{2}\,.$$

Важное наблюдение

Лемма

MAX-kCSP$_q(1-\alpha, 1) \in \mathrm{PCP}(\log n, 1)$. Здесь $\alpha = O(1)$.
Проверки неадаптивные.

Алгоритм проверки (ожидает доказательство в виде списка значений переменных в присваивании):

1. выбираем $C = \lceil -\log_2(1-\alpha) \rceil$ ограничений случайно и равновероятно;
2. запрашиваем значения соответствующих переменных, всего $O(kC \log q) = O(1)$ битов запрошено;
3. если все ограничения выполнены, то ответ «да», иначе «нет».

Если доказательство корректное, то ответ положительный всегда.
Если $\mathrm{UNSAT}(G) > \alpha$, то выполнено не более $1-\alpha$ доли ограничений на любом доказательстве; вероятность положительного ответа не превосходит

$$(1-\alpha)^C < \frac{1}{2}\,.$$

Теорема

В предположении усиленной PCP теоремы задача MAX-3SAT$(1-\varepsilon, 1)$ NP-трудна для некоторой константы ε.

Доказательство:

1. Рассмотрим неадаптивный $\mathrm{PCP}(\log n, 1)$ алгоритм для полного языка $L \in \mathrm{NP}$.
2. Для входа w длины n обозначим через $x_1, \ldots, x_m$ биты вероятностно проверямого доказательства, $m = \mathrm{poly}(n)$.
3. Пусть PCP-алгоритм делает k запросов и его ответ определяется булевой функцией f от k аргументов.
4. Всего разных вариантов запросов $\mathrm{poly}(n) = \exp O(\log n)$.
5. Получаем сводимость к задаче MAX-kCSP$_2(1/2, 1)$, ограничения задаются условием $f(x_{i_1}, \ldots, x_{i_k}) = 1$.

Теорема

В предположении усиленной PCP теоремы задача MAX-3SAT$(1-\varepsilon, 1)$ NP-трудна для некоторой константы ε.

Доказательство:

1. Рассмотрим неадаптивный $\mathrm{PCP}(\log n, 1)$ алгоритм для полного языка $L \in \mathrm{NP}$.
2. Для входа w длины n обозначим через $x_1, \ldots, x_m$ биты вероятностно проверямого доказательства, $m = \mathrm{poly}(n)$.
3. Пусть PCP-алгоритм делает k запросов и его ответ определяется булевой функцией f от k аргументов.
4. Всего разных вариантов запросов $\mathrm{poly}(n) = \exp O(\log n)$.
5. Получаем сводимость к задаче MAX-kCSP$_2(1/2, 1)$, ограничения задаются условием $f(x_{i_1}, \ldots, x_{i_k}) = 1$.

Теорема

В предположении усиленной PCP теоремы задача MAX-3SAT$(1-\varepsilon, 1)$ NP-трудна для некоторой константы ε.

Доказательство:

1. Рассмотрим неадаптивный $\mathrm{PCP}(\log n, 1)$ алгоритм для полного языка $L \in \mathrm{NP}$.
2. Для входа w длины n обозначим через $x_1, \dots, x_m$ биты вероятностно проверямого доказательства, $m = \mathrm{poly}(n)$.
3. Пусть PCP-алгоритм делает k запросов и его ответ определяется булевой функцией f от k аргументов.
4. Всего разных вариантов запросов $\mathrm{poly}(n) = \exp O(\log n)$.
5. Получаем сводимость к задаче MAX-kCSP$_2(1/2, 1)$, ограничения задаются условием $f(x_{i_1}, \dots, x_{i_k}) = 1$.

Задача выполнимости: трудность приближения

Теорема

В предположении усиленной PCP теоремы задача MAX-3SAT$(1-\varepsilon, 1)$ NP-трудна для некоторой константы ε.

Доказательство:

1. Рассмотрим неадаптивный $\mathrm{PCP}(\log n, 1)$ алгоритм для полного языка $L \in \mathrm{NP}$.
2. Для входа w длины n обозначим через $x_1, \ldots, x_m$ биты вероятностно проверямого доказательства, $m = \mathrm{poly}(n)$.
3. Пусть PCP-алгоритм делает k запросов и его ответ определяется булевой функцией f от k аргументов.
4. Всего разных вариантов запросов $\mathrm{poly}(n) = \exp O(\log n)$.
5. Получаем сводимость к задаче MAX-kCSP$_2(1/2, 1)$, ограничения задаются условием $f(x_{i_1}, \ldots, x_{i_k}) = 1$.

Задача выполнимости: трудность приближения

Теорема

В предположении усиленной PCP теоремы задача MAX-3SAT$(1-\varepsilon, 1)$ NP-трудна для некоторой константы ε.

Доказательство:

1. Рассмотрим неадаптивный $\mathrm{PCP}(\log n, 1)$ алгоритм для полного языка $L \in \mathrm{NP}$.
2. Для входа w длины n обозначим через $x_1, \dots, x_m$ биты вероятностно проверямого доказательства, $m = \mathrm{poly}(n)$.
3. Пусть PCP-алгоритм делает k запросов и его ответ определяется булевой функцией f от k аргументов.
4. Всего разных вариантов запросов $\mathrm{poly}(n) = \exp O(\log n)$.
5. Получаем сводимость к задаче MAX-kCSP$_2(1/2, 1)$, ограничения задаются условием $f(x_{i_1}, \dots, x_{i_k}) = 1$.

Задача выполнимости: трудность приближения

Теорема

В предположении усиленной PCP теоремы задача MAX-3SAT$(1-\varepsilon, 1)$ NP-трудна для некоторой константы ε.

Доказательство:

1. Рассмотрим неадаптивный $\mathrm{PCP}(\log n, 1)$ алгоритм для полного языка $L \in \mathrm{NP}$.
2. Для входа w длины n обозначим через $x_1, \dots, x_m$ биты вероятностно проверямого доказательства, $m = \mathrm{poly}(n)$.
3. Пусть PCP-алгоритм делает k запросов и его ответ определяется булевой функцией f от k аргументов.
4. Всего разных вариантов запросов $\mathrm{poly}(n) = \exp O(\log n)$.
5. Получаем сводимость к задаче MAX-kCSP$_2(1/2, 1)$, ограничения задаются условием $f(x_{i_1}, \dots, x_{i_k}) = 1$.

Задача выполнимости: трудность приближения (2)

Теорема

В предположении усиленной PCP теоремы задача MAX-3SAT$(1-\varepsilon, 1)$ NP-трудна для некоторой константы ε.

Доказательство (продолжение):

6. Разложим f в совершенную КНФ C_f, каждый дизъюнкт которой содержит k литералов. Общее число дизъюнктов 2^k.
7. Выполнимость C_f равносильна выполнимости 3-КНФ T_f, которая получается заменой каждого дизъюнкта

$$\ell_1 \vee \ell_2 \vee \dots \vee \ell_k$$

на 3-КНФ

$$(x \vee \ell_1 \vee t_1)(\neg t_1 \vee \ell_2 \vee t_2) \dots (\neg t_{k-1} \vee \ell_k \vee x) \bigwedge_{\alpha,\beta} (\neg x \vee y^\alpha \vee z^\beta)$$

со вспомогательными переменными t_i, x, y, z.

Задача выполнимости: трудность приближения (2)

Теорема

В предположении усиленной PCP теоремы задача MAX-3SAT$(1-\varepsilon, 1)$ NP-трудна для некоторой константы ε.

Доказательство (продолжение):

6. Разложим f в совершенную КНФ C_f, каждый дизъюнкт которой содержит k литералов. Общее число дизъюнктов 2^k.
7. Выполнимость C_f равносильна выполнимости 3-КНФ T_f, которая получается заменой каждого дизъюнкта

$$\ell_1 \vee \ell_2 \vee \cdots \vee \ell_k$$

на 3-КНФ

$$(x \vee \ell_1 \vee t_1)(\neg t_1 \vee \ell_2 \vee t_2)\ldots(\neg t_{k-1} \vee \ell_k \vee x) \bigwedge_{\alpha,\beta}(\neg x \vee y^{\alpha} \vee z^{\beta})$$

со вспомогательными переменными t_i, x, y, z.

Задача выполнимости: трудность приближения (3)

Теорема

В предположении усиленной PCP теоремы задача MAX-3SAT$(1-\varepsilon, 1)$ NP-трудна для некоторой константы ε.

Доказательство (продолжение):

8. Количество ограничений увеличилось в $(k+4)2^k$ раз. Если нарушается хотя бы половина исходных ограничений, то в T_f нарушается хотя бы $\dfrac{1}{(k+4)2^{k+1}}$ доля ограничений.
9. Получили сводимость $w \mapsto T_f$ задачи L к задаче MAX-3SAT$\left(1-\dfrac{1}{(k+4)2^{k+1}}, 1\right)$.

Задача выполнимости: трудность приближения (3)

Теорема

В предположении усиленной PCP теоремы задача MAX-3SAT$(1-\varepsilon, 1)$ NP-трудна для некоторой константы ε.

Доказательство (продолжение):

8. Количество ограничений увеличилось в $(k+4)2^k$ раз. Если нарушается хотя бы половина исходных ограничений, то в T_f нарушается хотя бы $\dfrac{1}{(k+4)2^{k+1}}$ доля ограничений.
9. Получили сводимость $w \mapsto T_f$ задачи L к задаче MAX-3SAT$\left(1 - \dfrac{1}{(k+4)2^{k+1}}, 1\right)$.

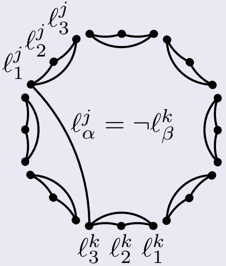
MAX-IND: константная неаппроксимируемость

Сводимость MAX-3SAT(a, 1) к MAX-IND(am, m)

Поставим в соответстие 3КНФ

$$C = \bigwedge_{j=1}^{m} (\ell_1^j \vee \ell_2^j \vee \ell_3^j)$$

граф G на $3m$ вершинах, индексированных ℓ_α^j:



Рёбра между тройками соединяют несовместные литералы

Независимое множество размера s в графе G существует тогда и только тогда, когда в C можно выполнить s дизъюнктов.

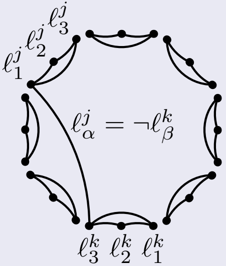
MAX-IND: константная неаппроксимируемость

Сводимость MAX-3SAT$(a, 1)$ к MAX-IND(am, m)

Поставим в соответстие 3КНФ

$$C = \bigwedge_{j=1}^{m} (\ell_1^j \vee \ell_2^j \vee \ell_3^j)$$

граф G на $3m$ вершинах, индексированных ℓ_α^j:



Рёбра между тройками соединяют несовместные литералы

Независимое множество размера s в графе G существует тогда и только тогда, когда в C можно выполнить s дизъюнктов.

MAX-IND: неаппроксимируемость с любой константой

Полный граф маршрутов $G^{\langle t\rangle}$

$V(G^{\langle t\rangle})$ — все последовательности из $V(G)$ длины t.
Последовательности $(v_1, \dots, v_t)$ и $(u_1, \dots, u_t)$ соединены ребром в графе $G^{\langle t\rangle}$, если в $\{v_i\} \cup \{u_j\}$ есть хотя бы одно ребро в графе G.

Утверждение

Пусть $\alpha(G)$ — число независимости. Тогда $\alpha(G^{\langle t\rangle}) = \alpha(G)^t$.

Достаточно заметить, что максимальное независимое множество в $G^{\langle t\rangle}$ образовано всеми последовательностями, лежащими в максимальном независимом множестве в G.

Получаем сводимость MAX-IND с зазором a к MAX-IND с зазором a^t.

MAX-IND: неаппроксимируемость с любой константой

Полный граф маршрутов $G^{\langle t\rangle}$

$V(G^{\langle t\rangle})$ — все последовательности из $V(G)$ длины t.
Последовательности $(v_1, \ldots, v_t)$ и $(u_1, \ldots, u_t)$ соединены ребром в графе $G^{\langle t\rangle}$, если в $\{v_i\} \cup \{u_j\}$ есть хотя бы одно ребро в графе G.

Утверждение

Пусть $\alpha(G)$ — число независимости. Тогда $\alpha(G^{\langle t\rangle}) = \alpha(G)^t$.

Достаточно заметить, что максимальное независимое множество в $G^{\langle t\rangle}$ образовано всеми последовательностями, лежащими в максимальном независимом множестве в G.

Получаем сводимость MAX-IND с зазором a к MAX-IND с зазором a^t.

MAX-IND: неаппроксимируемость с любой константой

Полный граф маршрутов $G^{\langle t \rangle}$

$V(G^{\langle t \rangle})$ — все последовательности из $V(G)$ длины t.
Последовательности $(v_1, \dots, v_t)$ и $(u_1, \dots, u_t)$ соединены ребром в графе $G^{\langle t \rangle}$, если в $\{v_i\} \cup \{u_j\}$ есть хотя бы одно ребро в графе G.

Утверждение

Пусть $\alpha(G)$ — число независимости. Тогда $\alpha(G^{\langle t \rangle}) = \alpha(G)^t$.

Достаточно заметить, что максимальное независимое множество в $G^{\langle t \rangle}$ образовано всеми последовательностями, лежащими в максимальном независимом множестве в G.

Получаем сводимость MAX-IND с зазором a к MAX-IND с зазором a^t.

MAX-IND: неаппроксимируемость с любой константой

Полный граф маршрутов $G^{\langle t\rangle}$

$V(G^{\langle t\rangle})$ — все последовательности из $V(G)$ длины t.
Последовательности $(v_1, \ldots, v_t)$ и $(u_1, \ldots, u_t)$ соединены ребром в графе $G^{\langle t\rangle}$, если в $\{v_i\} \cup \{u_j\}$ есть хотя бы одно ребро в графе G.

Утверждение

Пусть $\alpha(G)$ — число независимости. Тогда $\alpha(G^{\langle t\rangle}) = \alpha(G)^t$.

Достаточно заметить, что максимальное независимое множество в $G^{\langle t\rangle}$ образовано всеми последовательностями, лежащими в максимальном независимом множестве в G.

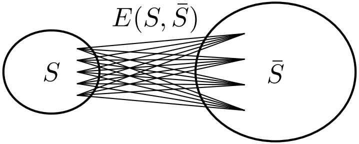
Получаем сводимость MAX-IND с зазором a к MAX-IND с зазором a^t.

План

Реберные экспандеры

Определение коэффициента реберного расширения $G(V, E)$

$$h(G) = \min_{S:|S|<|V|/2} \frac{E(S,\bar{S})}{|S|} .$$



Теорема (эффективное семейство экспандеров)

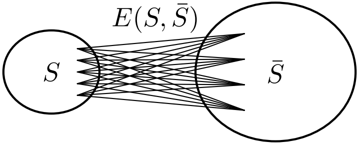
Существуют константы d_0, h_0 и полиномиальный по n алгоритм $n \mapsto G_n$ такие, что

- граф G_n содержит n вершин;
- каждая вершина графа G_n имеет степень d_0;
- $h(G_n) \geq h_0$.

Реберные экспандеры

Определение коэффициента реберного расширения $G(V, E)$

$$h(G) = \min_{S:|S|<|V|/2} \frac{E(S,\bar{S})}{|S|}\,.$$



Теорема (эффективное семейство экспандеров)

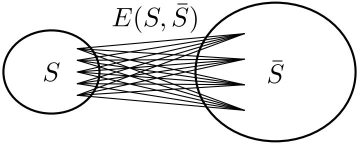
Существуют константы d_0, h_0 и полиномиальный по n алгоритм $n \mapsto G_n$ такие, что

- граф G_n содержит n вершин;
- каждая вершина графа G_n имеет степень d_0;
- $h(G_n) \geqslant h_0$.

Реберные экспандеры

Определение коэффициента реберного расширения $G(V, E)$

$$h(G) = \min_{S:|S|<|V|/2} \frac{E(S, \bar{S})}{|S|} .$$



Теорема (эффективное семейство экспандеров)

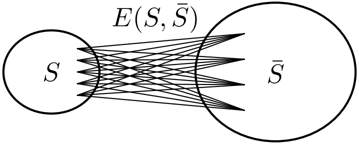
Существуют константы d_0, h_0 и полиномиальный по n алгоритм $n \mapsto G_n$ такие, что

- граф G_n содержит n вершин;
- каждая вершина графа G_n имеет степень d_0;
- $h(G_n) \geqslant h_0$.

Реберные экспандеры

Определение коэффициента реберного расширения $G(V, E)$

$$h(G) = \min_{S:|S|<|V|/2} \frac{E(S,\bar{S})}{|S|}.$$



Теорема (эффективное семейство экспандеров)

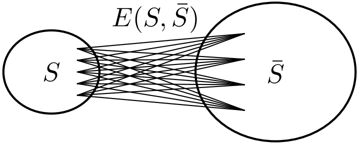
Существуют константы d_0, h_0 и полиномиальный по n алгоритм $n \mapsto G_n$ такие, что

- граф G_n содержит n вершин;
- каждая вершина графа G_n имеет степень d_0;
- $h(G_n) \geqslant h_0$.

Реберные экспандеры

Определение коэффициента реберного расширения $G(V, E)$

$$h(G) = \min_{S : |S| < |V|/2} \frac{E(S, \bar{S})}{|S|}.$$



Теорема (эффективное семейство экспандеров)

Существуют константы d_0, h_0 и полиномиальный по n алгоритм $n \mapsto G_n$ такие, что

- граф G_n содержит n вершин;
- каждая вершина графа G_n имеет степень d_0;
- $h(G_n) \geqslant h_0$.

Алгебраические экспандеры

- Пусть G — d-регулярный граф, $A(G)$ — матрица смежности вершин.
- $A(G)$ — симметрическая, поэтому собственные числа $A(G)$ действительные. Упорядочим их по величине модуля.
- Наибольшее по модулю собственное число равно d.
- Второй по величине модуль собственного числа $A(G)$ обозначим $\lambda(G)$.
- Если $\lambda(G)/d < \alpha$, то граф является алгебраическим (α, d)-экспандером.

Важное свойство (неформально)

Блуждание по алгебраическому экспандеру быстро сходится к равномерному распределению.

Алгебраические экспандеры

- Пусть G — d-регулярный граф, $A(G)$ — матрица смежности вершин.
- $A(G)$ — симметрическая, поэтому собственные числа $A(G)$ действительные. Упорядочим их по величине модуля.
- Наибольшее по модулю собственное число равно d.
- Второй по величине модуль собственного числа $A(G)$ обозначим $\lambda(G)$.
- Если $\lambda(G)/d < \alpha$, то граф является алгебраическим (α, d)-экспандером.

Важное свойство (неформально)

Блуждание по алгебраическому экспандеру быстро сходится к равномерному распределению.

Алгебраические экспандеры

- Пусть G — d-регулярный граф, $A(G)$ — матрица смежности вершин.
- $A(G)$ — симметрическая, поэтому собственные числа $A(G)$ действительные. Упорядочим их по величине модуля.
- Наибольшее по модулю собственное число равно d.
- Второй по величине модуль собственного числа $A(G)$ обозначим $\lambda(G)$.
- Если $\lambda(G)/d < \alpha$, то граф является алгебраическим (α, d)-экспандером.

Важное свойство (неформально)

Блуждание по алгебраическому экспандеру быстро сходится к равномерному распределению.

Алгебраические экспандеры

- Пусть G — d-регулярный граф, $A(G)$ — матрица смежности вершин.
- $A(G)$ — симметрическая, поэтому собственные числа $A(G)$ действительные. Упорядочим их по величине модуля.
- Наибольшее по модулю собственное число равно d.
- Второй по величине модуль собственного числа $A(G)$ обозначим $\lambda(G)$.
- Если $\lambda(G)/d < \alpha$, то граф является алгебраическим (α, d)-экспандером.

Важное свойство (неформально)

Блуждание по алгебраическому экспандеру быстро сходится к равномерному распределению.

Алгебраические экспандеры

- Пусть G — d-регулярный граф, $A(G)$ — матрица смежности вершин.
- $A(G)$ — симметрическая, поэтому собственные числа $A(G)$ действительные. Упорядочим их по величине модуля.
- Наибольшее по модулю собственное число равно d.
- Второй по величине модуль собственного числа $A(G)$ обозначим $\lambda(G)$.
- Если $\lambda(G)/d < \alpha$, то граф является алгебраическим (α, d)-экспандером.

Важное свойство (неформально)

Блуждание по алгебраическому экспандеру быстро сходится к равномерному распределению.

Алгебраические экспандеры

- Пусть G — d-регулярный граф, $A(G)$ — матрица смежности вершин.
- $A(G)$ — симметрическая, поэтому собственные числа $A(G)$ действительные. Упорядочим их по величине модуля.
- Наибольшее по модулю собственное число равно d.
- Второй по величине модуль собственного числа $A(G)$ обозначим $\lambda(G)$.
- Если $\lambda(G)/d < \alpha$, то граф является алгебраическим (α, d)-экспандером.

Важное свойство (неформально)

Блуждание по алгебраическому экспандеру быстро сходится к равномерному распределению.

Алгебраические и реберные экспандеры — одно и то же

Теорема (реберный $\Rightarrow$ алгебраический)

Пусть G — d-регулярный граф, в каждой вершине которого есть петля, $\lambda_2(G)$ — второе собственное число матрицы смежности G. Тогда

$$\lambda_2(G) \leqslant d - \frac{h(G)^2}{2d}, \qquad \lambda(G) \leqslant \max\left(d-2, d - \frac{h(G)^2}{2d}\right).$$

Теорема (алгебраический $\Rightarrow$ реберный)

$$h(G) \geqslant \frac{d-\lambda(G)}{2}.$$

Следствие

Для некоторых констант $\lambda_0 < 1$, d_0 существует эффективно построимое семейство (λ_0, d_0)-экспандеров.

Алгебраические и реберные экспандеры — одно и то же

Теорема (реберный $\Rightarrow$ алгебраический)

Пусть G — d-регулярный граф, в каждой вершине которого есть петля, $\lambda_2(G)$ — второе собственное число матрицы смежности G. Тогда

$$\lambda_2(G) \leqslant d - \frac{h(G)^2}{2d}, \qquad \lambda(G) \leqslant \max\left(d-2, d-\frac{h(G)^2}{2d}\right).$$

Теорема (алгебраический $\Rightarrow$ реберный)

$$h(G) \geqslant \frac{d-\lambda(G)}{2}.$$

Следствие

Для некоторых констант $\lambda_0 < 1$, d_0 существует эффективно построимое семейство (λ_0, d_0)-экспандеров.

Алгебраические и реберные экспандеры — одно и то же

Теорема (реберный $\Rightarrow$ алгебраический)

Пусть G — d-регулярный граф, в каждой вершине которого есть петля, $\lambda_2(G)$ — второе собственное число матрицы смежности G. Тогда

$$\lambda_2(G) \leqslant d - \frac{h(G)^2}{2d}, \qquad \lambda(G) \leqslant \max\left(d-2, d - \frac{h(G)^2}{2d}\right) .$$

Теорема (алгебраический $\Rightarrow$ реберный)

$$h(G) \geqslant \frac{d - \lambda(G)}{2} .$$

Следствие

Для некоторых констант $\lambda_0 < 1$, d_0 существует эффективно построимое семейство (λ_0, d_0)-экспандеров.

Улучшение экспандера

Определение

Степень G^t графа G: матрица смежности $A(G^t) = (A(G))^t$.

Наблюдение

$\lambda(G^t) = (\lambda(G))^t$.

Следствие

Для любого $0 < \lambda < \lambda_0$ существует эффективно построимое семейство (λ, d_0)-экспандеров.

Улучшение экспандера

Определение

Степень G^t графа G: матрица смежности $A(G^t) = (A(G))^t$.

Наблюдение

$\lambda(G^t) = \left(\lambda(G)\right)^t$.

Следствие

Для любого $0 < \lambda < \lambda_0$ существует эффективно построимое семейство (λ, d_0)-экспандеров.

Улучшение экспандера

Определение

Степень G^t графа G: матрица смежности $A(G^t) = (A(G))^t$.

Наблюдение

$\lambda(G^t) = \left(\lambda(G)\right)^t$.

Следствие

Для любого $0 < \lambda < \lambda_0$ существует эффективно построимое семейство (λ, d_0)-экспандеров.

Случайные блуждания по экспандеру

Теорема

G — граф с множеством вершин V, $\lambda = \lambda(G)$.
$S \subset V$, $|S| = \beta|V|$ — подмножество вершин.
Вероятность $p(S, t)$ того, что случайный маршрут по экспандеру длины t целиком лежит в множестве S оценивается как

$$(\beta - \lambda)^{t+1} \leqslant p(S, t) \leqslant (\beta + \lambda)^{t+1}.$$

Усечение графа маршрутов

Усеченный граф маршрутов $G_H^{\langle t\rangle}$

H — (λ, d)-экспандер на множестве вершин графа G.
Вершины $G_H^{\langle t\rangle}$ — маршруты по экспандеру H длины t. Рёбра определяются так же, как для графа маршрутов: маршруты $\tau_1 = (v_1, \dots, v_t)$ и $\tau_2 = (u_1, \dots, u_t)$ соединены ребром в графе $G_H^{\langle t\rangle}$, если в $\{v_i\} \cup \{u_j\}$ есть хотя бы одно ребро в графе G.
Количество вершин в графе $G_H^{\langle t\rangle}$ равно $V(G)d^t$.

1. Пусть T — независимое множество в $G_H^{\langle t\rangle}$. Тогда объединение вершин (графа G) по всем маршрутам из T — независимое множество в G.
2. Поэтому максимальное независимое множество в $G_H^{\langle t\rangle}$ состоит из маршрутов, целиком лежащих в некотором независимом множестве графа G.

Усечение графа маршрутов

Усеченный граф маршрутов $G_H^{\langle t\rangle}$

H — (λ, d)-экспандер на множестве вершин графа G.
Вершины $G_H^{\langle t\rangle}$ — маршруты по экспандеру H длины t. Рёбра определяются так же, как для графа маршрутов: маршруты $\tau_1 = (v_1, \ldots, v_t)$ и $\tau_2 = (u_1, \ldots, u_t)$ соединены ребром в графе $G_H^{\langle t\rangle}$, если в $\{v_i\} \cup \{u_j\}$ есть хотя бы одно ребро в графе G.
Количество вершин в графе $G_H^{\langle t\rangle}$ равно $V(G)d^t$.

1. Пусть T — независимое множество в $G_H^{\langle t\rangle}$. Тогда объединение вершин (графа G) по всем маршрутам из T — независимое множество в G.
2. Поэтому максимальное независимое множество в $G_H^{\langle t\rangle}$ состоит из маршрутов, целиком лежащих в некотором независимом множестве графа G.

Усечение графа маршрутов

Усеченный граф маршрутов $G_H^{\langle t\rangle}$

H — (λ, d)-экспандер на множестве вершин графа G.
Вершины $G_H^{\langle t\rangle}$ — маршруты по экспандеру H длины t. Рёбра определяются так же, как для графа маршрутов: маршруты $\tau_1 = (v_1, \ldots, v_t)$ и $\tau_2 = (u_1, \ldots, u_t)$ соединены ребром в графе $G_H^{\langle t\rangle}$, если в $\{v_i\} \cup \{u_j\}$ есть хотя бы одно ребро в графе G.
Количество вершин в графе $G_H^{\langle t\rangle}$ равно $V(G)d^t$.

1. Пусть T — независимое множество в $G_H^{\langle t\rangle}$. Тогда объединение вершин (графа G) по всем маршрутам из T — независимое множество в G.
2. Поэтому максимальное независимое множество в $G_H^{\langle t\rangle}$ состоит из маршрутов, целиком лежащих в некотором независимом множестве графа G.

Неаппроксимируемость MAX-IND

Обозначим $\tilde{\alpha}(G) = \dfrac{\alpha(G)}{V(G)}$.

Из теоремы о блуждании получаем оценку

$$(\tilde{\alpha}(G) - \lambda)^t \leqslant \tilde{\alpha}(G_H^{\langle t\rangle}) \leqslant (\tilde{\alpha}(G) + \lambda)^t.$$

Оценим улучшение зазора при отображении $G \mapsto G_H^{\langle t\rangle}$:

1. Был зазор $((1-\varepsilon)V/3, V/3)$. Выберем $\lambda < \varepsilon/6$, $t = \log V$.
2. Для $G_H^{\langle t\rangle}$ зазор
$$\left(\left(\frac{1-\varepsilon}{3} + \lambda\right)^t V(G)d^t, \quad \left(\frac{1}{3} - \lambda\right)^t V(G)d^t\right)$$
3. Получаем неаппроксимируемость с погрешностью
$$\left(\frac{1-\varepsilon+3\lambda}{1-3\lambda}\right)^t > \gamma^t > V^{-c}, \quad c \approx -\log\gamma, \quad \gamma < 1.$$

Неаппроксимируемость MAX-IND

Обозначим $\tilde{\alpha}(G) = \dfrac{\alpha(G)}{V(G)}$.

Из теоремы о блуждании получаем оценку

$$(\tilde{\alpha}(G) - \lambda)^t \leqslant \tilde{\alpha}(G_H^{\langle t \rangle}) \leqslant (\tilde{\alpha}(G) + \lambda)^t.$$

Оценим улучшение зазора при отображении $G \mapsto G_H^{\langle t \rangle}$:

1. Был зазор $((1 - \varepsilon)V/3, V/3)$. Выберем $\lambda < \varepsilon/6$, $t = \log V$.
2. Для $G_H^{\langle t \rangle}$ зазор

$$\left(\left(\frac{1-\varepsilon}{3} + \lambda\right)^t V(G)d^t, \quad \left(\frac{1}{3} - \lambda\right)^t V(G)d^t\right)$$

3. Получаем неаппроксимируемость с погрешностью

$$\left(\frac{1-\varepsilon+3\lambda}{1-3\lambda}\right)^t > \gamma^t > V^{-c}, \quad c \approx -\log\gamma, \quad \gamma < 1.$$

Неаппроксимируемость MAX-IND

Обозначим $\tilde{\alpha}(G) = \dfrac{\alpha(G)}{V(G)}$.

Из теоремы о блуждании получаем оценку

$$(\tilde{\alpha}(G) - \lambda)^t \leqslant \tilde{\alpha}(G_H^{\langle t\rangle}) \leqslant (\tilde{\alpha}(G) + \lambda)^t.$$

Оценим улучшение зазора при отображении $G \mapsto G_H^{\langle t\rangle}$:

1. Был зазор $((1-\varepsilon)V/3, V/3)$. Выберем $\lambda < \varepsilon/6$, $t = \log V$.
2. Для $G_H^{\langle t\rangle}$ зазор
$$\left(\left(\frac{1-\varepsilon}{3} + \lambda\right)^t V(G)d^t, \quad \left(\frac{1}{3} - \lambda\right)^t V(G)d^t\right)$$
3. Получаем неаппроксимируемость с погрешностью
$$\left(\frac{1-\varepsilon+3\lambda}{1-3\lambda}\right)^t > \gamma^t > V^{-c}, \quad c \approx -\log\gamma, \quad \gamma < 1.$$

Неаппроксимируемость MAX-IND

Обозначим $\tilde{\alpha}(G) = \dfrac{\alpha(G)}{V(G)}$.

Из теоремы о блуждании получаем оценку

$$(\tilde{\alpha}(G) - \lambda)^t \leqslant \tilde{\alpha}(G_H^{\langle t \rangle}) \leqslant (\tilde{\alpha}(G) + \lambda)^t.$$

Оценим улучшение зазора при отображении $G \mapsto G_H^{\langle t \rangle}$:

1. Был зазор $((1-\varepsilon)V/3, V/3)$. Выберем $\lambda < \varepsilon/6$, $t = \log V$.
2. Для $G_H^{\langle t \rangle}$ зазор
$$\left(\left(\frac{1-\varepsilon}{3} + \lambda\right)^t V(G)d^t, \quad \left(\frac{1}{3} - \lambda\right)^t V(G)d^t\right)$$
3. Получаем неаппроксимируемость с погрешностью
$$\left(\frac{1-\varepsilon+3\lambda}{1-3\lambda}\right)^t > \gamma^t > V^{-c}, \quad c \approx -\log\gamma, \quad \gamma < 1.$$

План

Общая схема доказательства (I. Dinur, 2005): сводимости между задачами выполнимости

Теорема (усиленная PCP)

$\mathrm{NP} \subseteq \mathrm{PCP}(\log n, 1)$, причем проверки неадаптивны.

Достаточно свести NP-полную задачу MAX-2CSP$_3(1-\mathrm{poly}(n^{-1}), 1)$ (3-раскраска) к задаче MAX-2CSP$_k(1-\varepsilon, 1)$ с зазором $\varepsilon = \Omega(1)$.
Для этого достаточно построить «сводимость удвоения зазора», т. е. такую сводимость между задачами MAX-2CSP$_k(1-\varepsilon', 1)$ и MAX-2CSP$_k(1-\varepsilon'', 1)$, что

1. На каждой итерации зазор увеличивается по крайней мере вдвое, если не превышает границу $\alpha = O(1)$.
2. Размер графа ограничений увеличивается линейно за итерацию.

Итерированное применение $O(\log n)$ сводимостей удвоения зазора выполняется за полиномиальное время и дает в итоге зазор $\Omega(1)$.

Общая схема доказательства (I. Dinur, 2005): сводимости между задачами выполнимости

Теорема (усиленная PCP)

$\mathrm{NP} \subseteq \mathrm{PCP}(\log n, 1)$, причем проверки неадаптивны.

Достаточно свести NP-полную задачу MAX-2CSP$_3(1 - \mathrm{poly}(n^{-1}), 1)$ (3-раскраска) к задаче MAX-2CSP$_k(1 - \varepsilon, 1)$ с зазором $\varepsilon = \Omega(1)$.

Для этого достаточно построить «сводимость удвоения зазора», т. е. такую сводимость между задачами MAX-2CSP$_k(1 - \varepsilon', 1)$ и MAX-2CSP$_k(1 - \varepsilon'', 1)$, что

1. На каждой итерации зазор увеличивается по крайней мере вдвое, если не превышает границу $\alpha = O(1)$.
2. Размер графа ограничений увеличивается [illegible]

Итерированное применение $O(\log n)$ сводимостей удвоения зазора выполняется за полиномиальное время и дает в итоге зазор $\Omega(1)$.

Общая схема доказательства (I. Dinur, 2005): сводимости между задачами выполнимости

Теорема (усиленная PCP)

$\mathrm{NP} \subseteq \mathrm{PCP}(\log n, 1)$, причем проверки неадаптивны.

Достаточно свести NP-полную задачу MAX-2CSP$_3(1 - \mathrm{poly}(n^{-1}), 1)$ (3-раскраска) к задаче MAX-2CSP$_k(1 - \varepsilon, 1)$ с зазором $\varepsilon = \Omega(1)$.
Для этого достаточно построить «сводимость удвоения зазора», т. е. такую сводимость между задачами MAX-2CSP$_k(1 - \varepsilon', 1)$ и MAX-2CSP$_k(1 - \varepsilon'', 1)$, что

1. На каждой итерации зазор увеличивается по крайней мере вдвое, если не превышает границу $\alpha = O(1)$.
2. Размер графа ограничений увеличивается линейно за итерацию.

Итерированное применение $O(\log n)$ сводимостей удвоения зазора выполняется за полиномиальное время и дает в итоге зазор $\Omega(1)$.

Общая схема доказательства (I. Dinur, 2005): сводимости между задачами выполнимости

Теорема (усиленная PCP)

$\mathrm{NP} \subseteq \mathrm{PCP}(\log n, 1)$, причем проверки неадаптивны.

Достаточно свести NP-полную задачу MAX-2CSP$_3(1-\mathrm{poly}(n^{-1}), 1)$ (3-раскраска) к задаче MAX-2CSP$_k(1-\varepsilon, 1)$ с зазором $\varepsilon = \Omega(1)$.

Для этого достаточно построить «сводимость удвоения зазора», т. е. такую сводимость между задачами MAX-2CSP$_k(1-\varepsilon', 1)$ и MAX-2CSP$_k(1-\varepsilon'', 1)$, что

1. На каждой итерации зазор увеличивается по крайней мере вдвое, если не превышает границу $\alpha = O(1)$.
2. Размер графа ограничений увеличивается линейно за итерацию.

Итерированное применение $O(\log n)$ сводимостей удвоения зазора выполняется за полиномиальное время и дает в итоге зазор $\Omega(1)$.

Общая схема доказательства (I. Dinur, 2005): сводимости между задачами выполнимости

Теорема (усиленная PCP)

$\mathrm{NP} \subseteq \mathrm{PCP}(\log n, 1)$, причем проверки неадаптивны.

Достаточно свести NP-полную задачу MAX-2CSP$_3(1 - \mathrm{poly}(n^{-1}), 1)$ (3-раскраска) к задаче MAX-2CSP$_k(1 - \varepsilon, 1)$ с зазором $\varepsilon = \Omega(1)$.
Для этого достаточно построить «сводимость удвоения зазора», т. е. такую сводимость между задачами MAX-2CSP$_k(1 - \varepsilon', 1)$ и MAX-2CSP$_k(1 - \varepsilon'', 1)$, что

1. На каждой итерации зазор увеличивается по крайней мере вдвое, если не превышает границу $\alpha = O(1)$.
2. Размер графа ограничений увеличивается линейно за итерацию.

Итерированное применение $O(\log n)$ сводимостей удвоения зазора выполняется за полиномиальное время и дает в итоге зазор $\Omega(1)$.

Общая схема доказательства (I. Dinur, 2005): сводимости между задачами выполнимости

Теорема (усиленная PCP)

$\mathrm{NP} \subseteq \mathrm{PCP}(\log n, 1)$, причем проверки неадаптивны.

Достаточно свести NP-полную задачу MAX-2CSP$_3(1 - \mathrm{poly}(n^{-1}), 1)$ (3-раскраска) к задаче MAX-2CSP$_k(1 - \varepsilon, 1)$ с зазором $\varepsilon = \Omega(1)$.
Для этого достаточно построить «сводимость удвоения зазора», т. е. такую сводимость между задачами MAX-2CSP$_k(1 - \varepsilon', 1)$ и MAX-2CSP$_k(1 - \varepsilon'', 1)$, что

1. На каждой итерации зазор увеличивается по крайней мере вдвое, если не превышает границу $\alpha = O(1)$.
2. Размер графа ограничений увеличивается линейно за итерацию.

Итерированное применение $O(\log n)$ сводимостей удвоения зазора выполняется за полиномиальное время и дает в итоге зазор $\Omega(1)$.

Сводимость удвоения зазора: общая картина

Состоит в композиции четырех преобразований

1. Улучшение графа ограничений (переход к экспандеру с большим количеством петель). Цена: уменьшение зазора.
2. Увеличение зазора за счет увеличения алфавита.
3. Уменьшение алфавита за счет увеличения арности и уменьшения зазора.
4. Возврат к 2-выполнимости. Цена: уменьшение зазора.

Сводимость удвоения зазора: общая картина

Состоит в композиции четырех преобразований

1. Улучшение графа ограничений (переход к экспандеру с большим количеством петель). Цена: уменьшение зазора.
2. Увеличение зазора за счет увеличения алфавита.
3. Уменьшение алфавита за счет увеличения арности и уменьшения зазора.
4. Возврат к 2-выполнимости. Цена: уменьшение зазора.

Сводимость удвоения зазора: общая картина

Состоит в композиции четырех преобразований

1. Улучшение графа ограничений (переход к экспандеру с большим количеством петель). Цена: уменьшение зазора.
2. Увеличение зазора за счет увеличения алфавита.
3. Уменьшение алфавита за счет увеличения арности и уменьшения зазора.
4. Возврат к 2-выполнимости. Цена: уменьшение зазора.

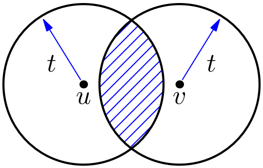
Сводимость удвоения зазора: общая картина

Состоит в композиции четырех преобразований

1. Улучшение графа ограничений (переход к экспандеру с большим количеством петель). Цена: уменьшение зазора.
2. Увеличение зазора за счет увеличения алфавита.
3. Уменьшение алфавита за счет увеличения арности и уменьшения зазора.
4. Возврат к 2-выполнимости. Цена: уменьшение зазора.

Увеличение зазора: основная идея

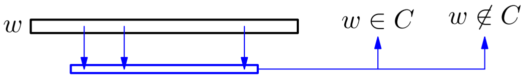
Проверка ограничений не для пары вершин, а для их окрестностей (описываем окрестность в расширенном алфавите).



Чтобы зазор увеличился, а размер графа изменился линейно, граф ограничений должен быть экспандером степени $O(1)$.

Уменьшение алфавита: основная идея

Уменьшение алфавита: использование локально-проверяемых кодов.



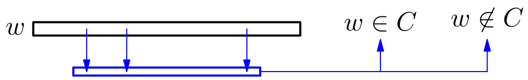
В конструкции Динур достаточно очень простого кода (так называемый длинный код).

Конструкция сводимости: для каждого ребра исходного графа ограничений записываем код пары символов в концах ребра. Код должен допускать следующие проверки после чтения константы битов кода:

1. корректность (действительно написано кодовое слово);
2. согласованность (если у ребер общая вершина, то записанные в них коды дают одно и то же значение в этой вершине);
3. выполнение ограничения исходной задачи на этом ребре.

Уменьшение алфавита: основная идея

Уменьшение алфавита: использование локально-проверяемых кодов.

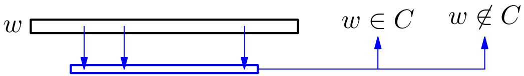


В конструкции Динур достаточно очень простого кода (так называемый длинный код).
Конструкция сводимости: для каждого ребра исходного графа ограничений записываем код пары символов в концах ребра. Код должен допускать следующие проверки после чтения константы битов кода:

1. корректность (действительно написано кодовое слово);
2. согласованность (если у ребер общая вершина, то записанные в них коды дают одно и то же значение в этой вершине);
3. выполнение ограничения исходной задачи на этом ребре.

Уменьшение алфавита: основная идея

Уменьшение алфавита: использование локально-проверяемых кодов.

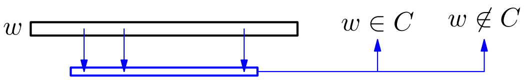


В конструкции Динур достаточно очень простого кода (так называемый длинный код).
Конструкция сводимости: для каждого ребра исходного графа ограничений записываем код пары символов в концах ребра. Код должен допускать следующие проверки после чтения константы битов кода:

1. корректность (действительно написано кодовое слово);
2. согласованность (если у ребер общая вершина, то записанные в них коды дают одно и то же значение в этой вершине);
3. выполнение ограничения исходной задачи на этом ребре.

Уменьшение алфавита: основная идея

Уменьшение алфавита: использование локально-проверяемых кодов.

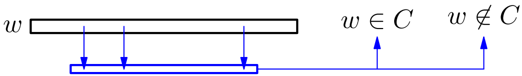


В конструкции Динур достаточно очень простого кода (так называемый длинный код).
Конструкция сводимости: для каждого ребра исходного графа ограничений записываем код пары символов в концах ребра. Код должен допускать следующие проверки после чтения константы битов кода:

1. корректность (действительно написано кодовое слово);
2. согласованность (если у ребер общая вершина, то записанные в них коды дают одно и то же значение в этой вершине);
3. выполнение ограничения исходной задачи на этом ребре.

Уменьшение алфавита: основная идея

Уменьшение алфавита: использование локально-проверяемых кодов.



В конструкции Динур достаточно очень простого кода (так называемый длинный код).

Конструкция сводимости: для каждого ребра исходного графа ограничений записываем код пары символов в концах ребра. Код должен допускать следующие проверки после чтения константы битов кода:

1. корректность (действительно написано кодовое слово);
2. согласованность (если у ребер общая вершина, то записанные в них коды дают одно и то же значение в этой вершине);
3. выполнение ограничения исходной задачи на этом ребре.