Модуль 2. Ускорение перебора

Лекция 7

Параметризованные алгоритмы.
Часть 1.

План лекции

1. Параметризованные задачи и алгоритмы. Классы XP, FPT.
2. Древовидный перебор ограниченной глубины (задача о вершинном покрытии).
3. Метод параметрической редукции данных.
4. Итерационной сжатие.

Как ускорить решение?

Полный перебор требует слишком много времени?

Варианты:

1) Эффективно решаемые частные случаи.

➔ *Параметризованные алгоритмы*

2) Приближённое решение (с оценкой точности или без).

Вершинное покрытие

Дан граф $G(V, E)$.

Вершинное покрытие:

$U \subseteq V$: $\forall\ e \in E \quad \exists\ u \in U$: ребро e инцидентно вершине u.

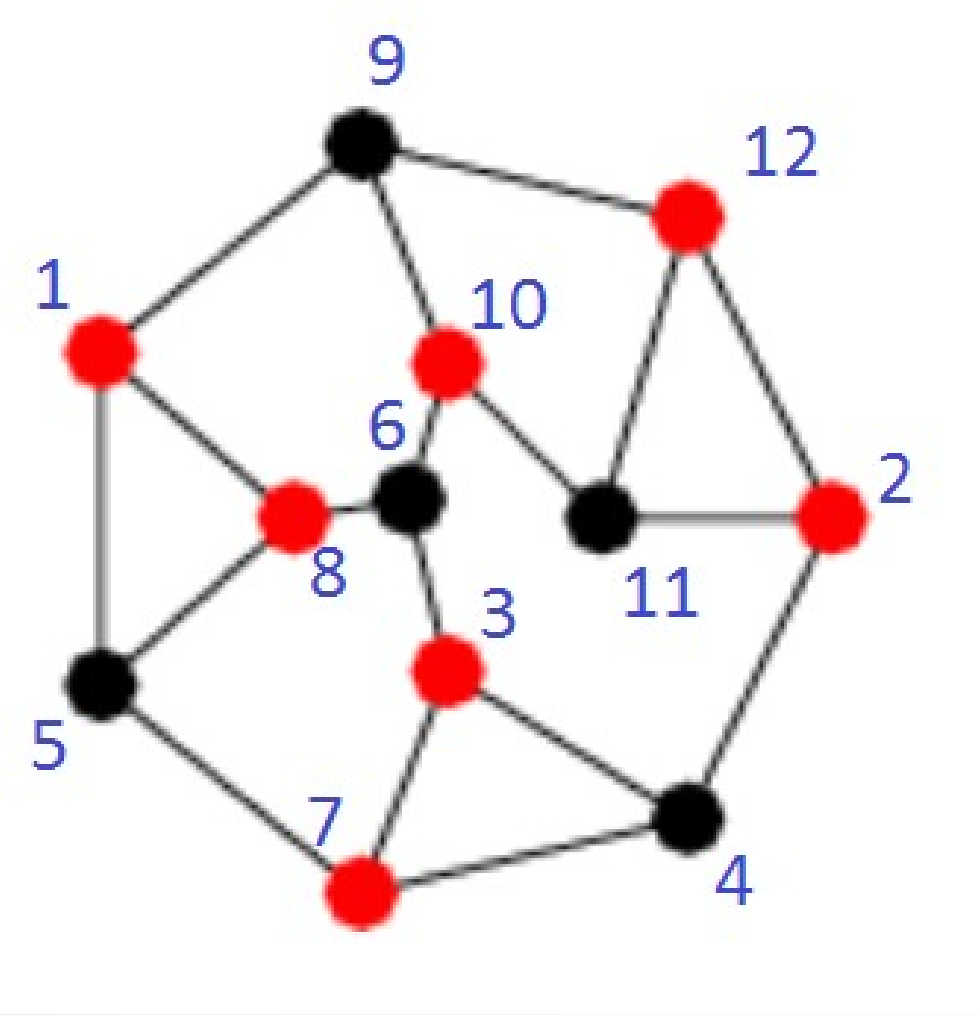
Оптимизационная задача: найти минимальное по мощности вершинное покрытие U.

Распознавательная задача: кроме графа, задано $k \in N$.
Определить, есть ли на графе вершинное покрытие мощности k или меньше.

Вычислительная задача: найти на графе вершинное покрытие мощности k или меньше.

Вершинное покрытие

http://mathworld.wolfram.com/VertexCover.html



Красные вершины образуют минимальное вершинное покрытие мощности 7

Параметризованные алгоритмы

Исходная задача:

Дано: $L \subseteq A^*, \quad x \in A^*.$

Найти: верно ли, что $x \in L$?

Параметризованная задача:

Дано: $L \subseteq A^* \times \mathbb{N}, \quad (x,k) \in A^* \times \mathbb{N}.$

Найти: верно ли, что $(x,k) \in L$?

Параметризованный алгоритм: алгоритм, для которого временная сложность = $T(n,k)$, $n=|x|$.

Параметризованные алгоритмы

Определение.

Параметризованная задача принадлежит классу **FPT** (= **F**ixed **P**arameter **T**ractable = разрешима при фиксированном параметре), если для неё существует параметризованный алгоритм с временной сложностью

$$T(n,k) = f(k)\cdot n^c$$

где f: $N \to N$ — вычислимая функция, c=const.

Параметризованные алгоритмы

Определение.

Параметризованная задача принадлежит классу **XP** (= slice-wise polynomial = кусочно полиномиальная по времени), если для неё существует параметризованный алгоритм с временной сложностью

$$T(n,k) = f(k)\cdot n^{g(k)}$$

где f,g: $N \to N$ — вычислимые функции.

Очевидно: $P \subseteq FPT \subseteq XP \subseteq NP$

Параметризованные алгоритмы

Что можно выбирать в качестве k?

- Явный параметр: задан непосредственно во входных данных
  - Размер решения (для задач минимизации)
- Неявный параметр: требуется дополнительно вычислить
  - Максимальная степень вершины в графе (maxdeg)
  - Род графа (степень «планарности» графа)
  - Древесная ширина графа (treewidth)

При разном выборе параметра одна и та же задача может иметь, а может и не иметь полиномиального (параметризованного) алгоритма!

Пример - задача о клике:

- k = размер клики $\Rightarrow$ CLIQUE $\notin$ FPT
- k = maxdeg(G) $\Rightarrow$ CLIQUE $\in$ FPT

Параметризованные алгоритмы

Разработка параметризованных алгоритмов — пока скорее искусство, чем готовая технология.

Но предложено множество методов. Наиболее часто используемые:

- Ограниченный древовидный перебор (bounded search trees).
- Параметрическая редукция данных (kernelization).
- Итерационное сжатие (iterative compression).
- Цветное кодирование (color coding).
- Вероятностное разделение (random separation).

Древовидный перебор

Потенциальные решения: подмножества $U \subseteq V$.

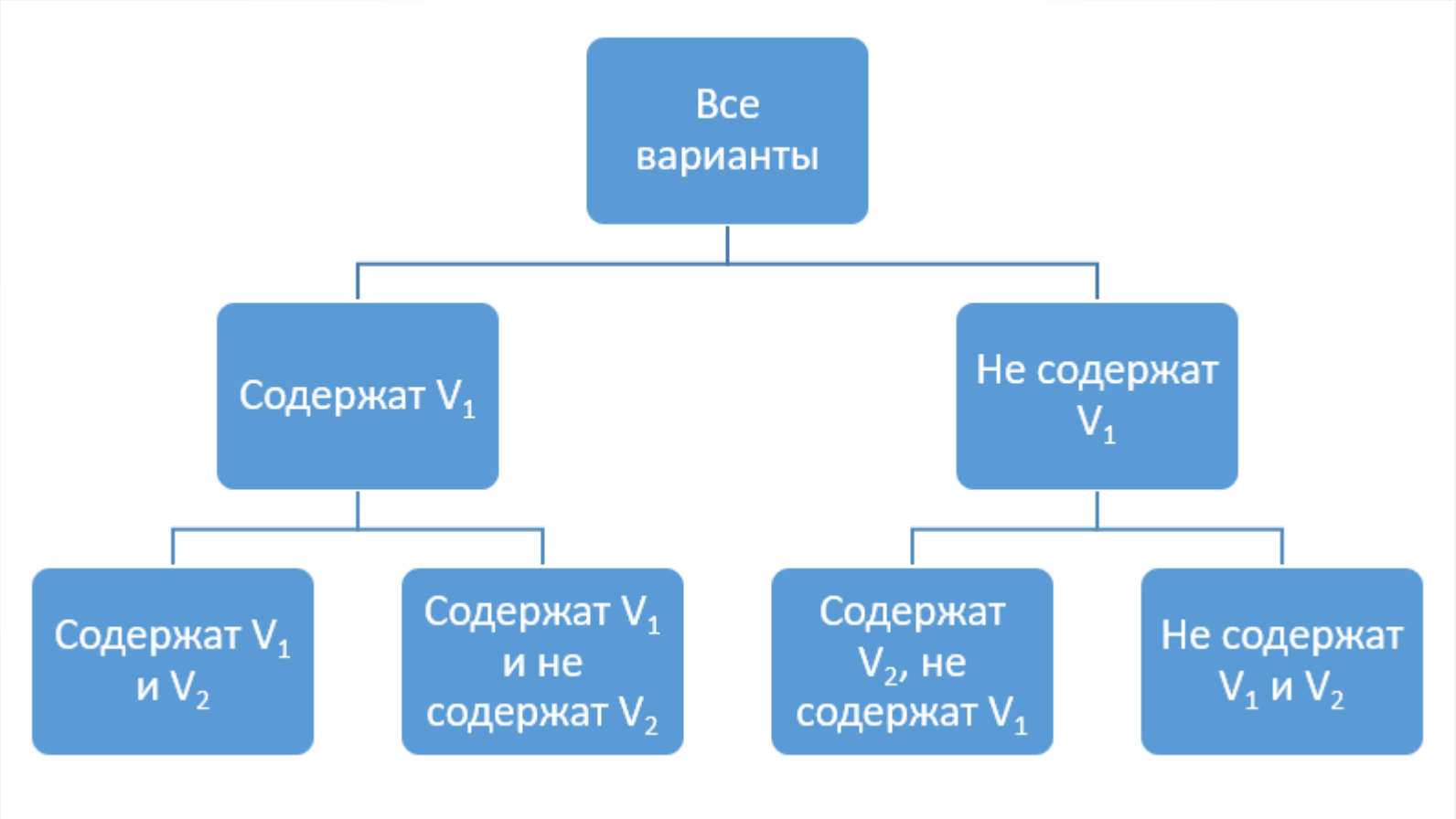
Выберем $v_1 \in V$.

Все потенциальные решения делятся на два класса:

- содержащие v_1;
- не содержащие v_1.

На последующих уровнях — расщепляем классы решений по другой вершине v_2. И так далее.

Древовидный перебор



Глубина дерева: $n = |V|$.

Сложность перебора: $O(2^n \cdot n^2)$.

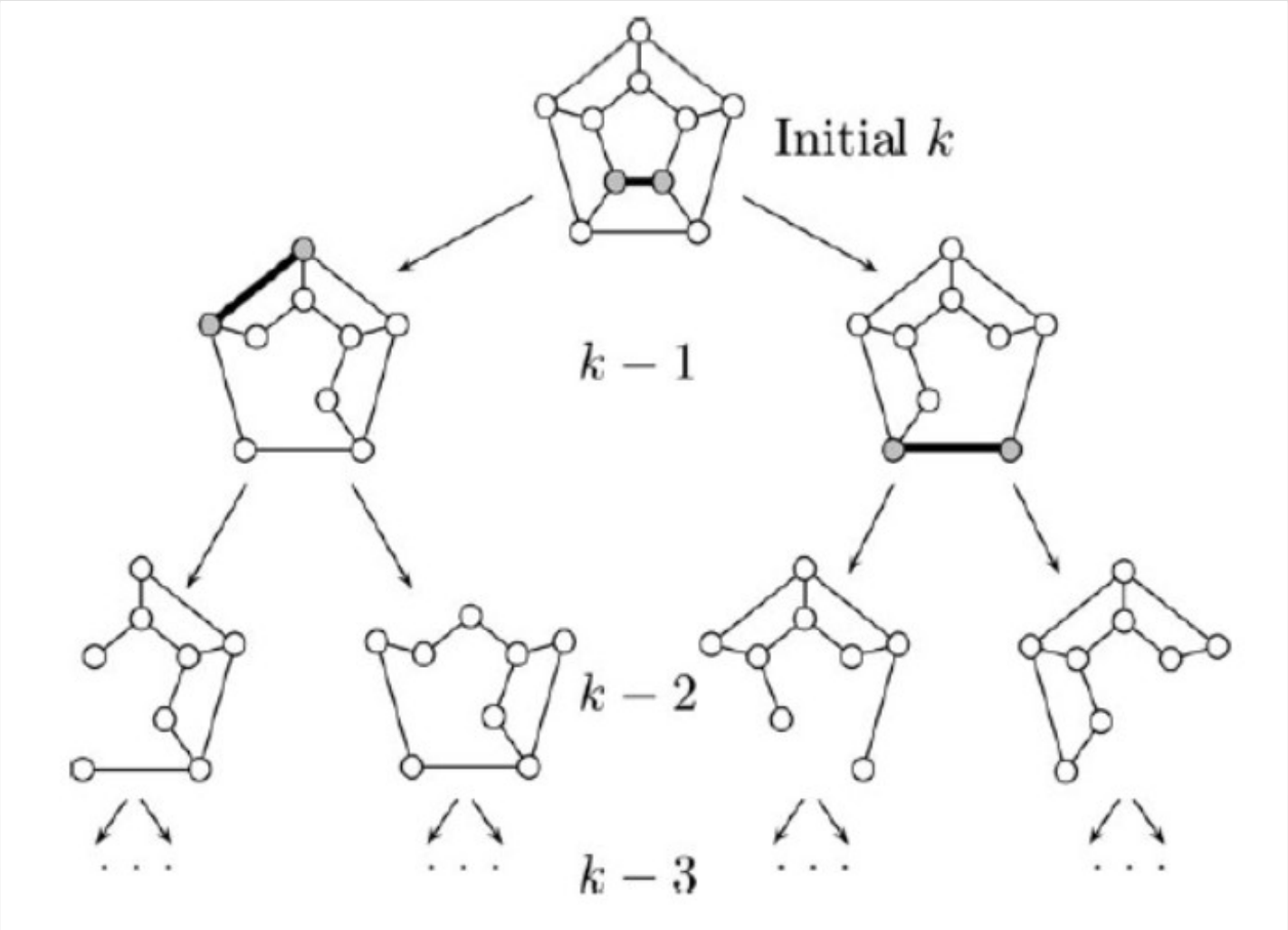
Древовидный перебор

Как можно сократить перебор?

Идея: найдём небольшое подмножество X = $\{x_1,...,x_m\}$, такое, что хотя бы одна вершина из X обязательно входит в покрытие. И разделим всё множество потенциальных покрытий на классы: в *i*-й класс входят покрытия, содержащие x_i.

Как выбрать X? Самый простой способ: выбрать ребро $e=(u,v)$, и $X:=\{u,v\}$.

Древовидный перебор



Сложность: $O(2^k \cdot n^2)$

Древовидный перебор

Вместо алгоритма сложности $O(2^n \cdot n^2)$

получили алгоритм сложности $O(2^k \cdot n^2)$.

При $k<<n$ получаем выигрыш в скорости.

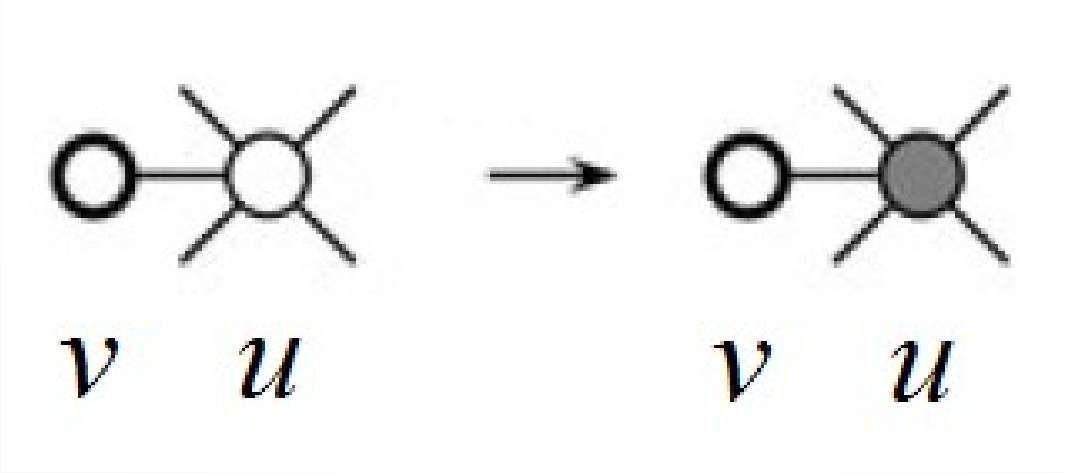
Можно ли сократить и время $O(2^k \cdot n^2)$?

Если мы ищем одно оптимальное решение, то можно!

Древовидный перебор

Наблюдение 1

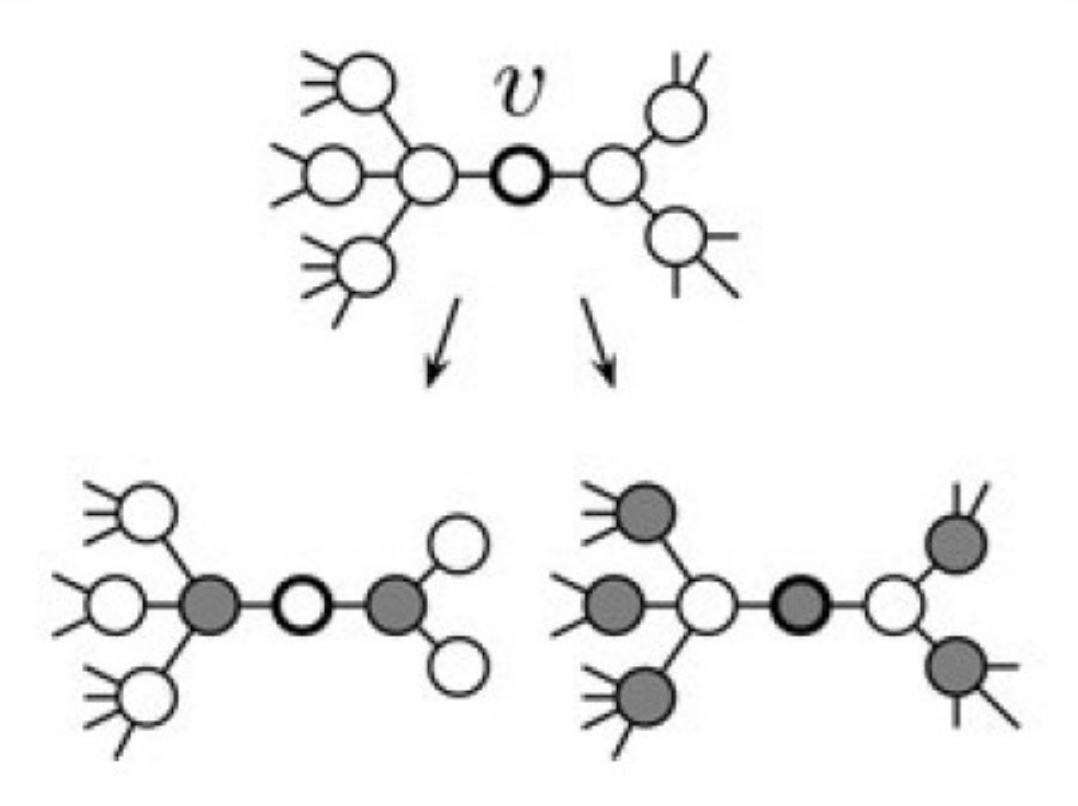
Если deg(*v*)=1, то смежную с ней вершину *u* можно сразу поместить в покрытие.



Древовидный перебор

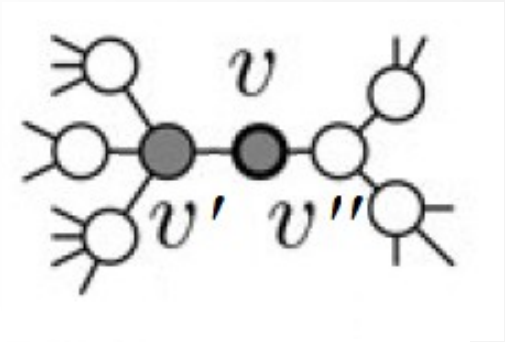
Наблюдение 2

Если deg(v)=2, то U содержит либо обоих соседей v, либо всех их соседей (включая v).

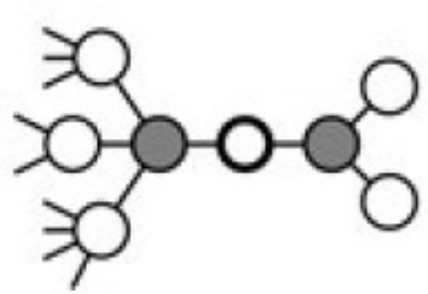


Древовидный перебор

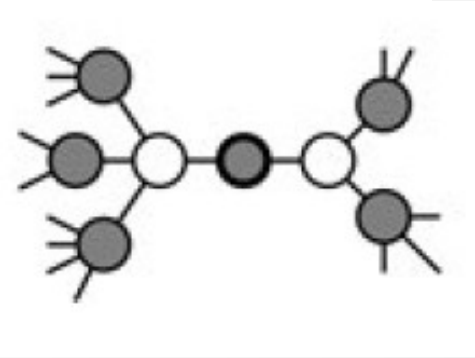
Действительно, пусть минимальное покрытие *U* содержит *v* и её соседа *v* '.



Если заменим *v* на *v* '',также получим минимальное покрытие (левая ветка).



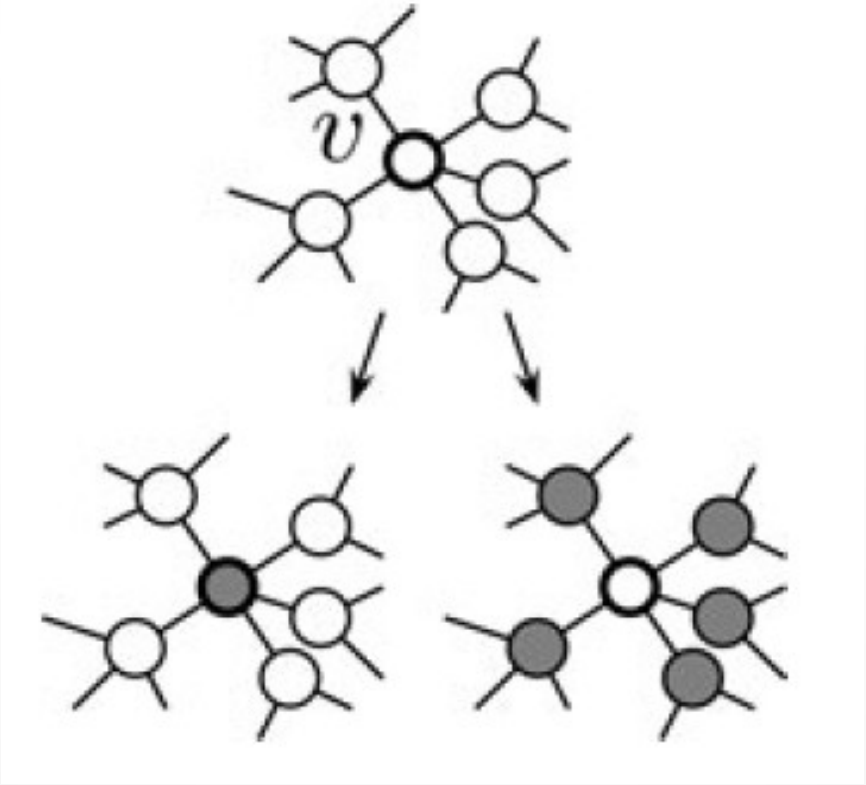
Следовательно: чтобы найти ещё меньшее покрытие, надо исключить *v* ' и *v* ''.



Древовидный перебор

Наблюдение 3

Если $\deg(v) \geq 3$, то U содержит либо вершину v, либо всех её соседей.



Древовидный перебор

На основе наблюдений (1) — (3) построим алгоритм ветвления.

В ситуации (1) — одна ветвь, в U добавляется 1 вершина.

В ситуации (2) — две ветви, по каждой в U добавляется минимум 2 вершины.

В ситуации (3) — две ветви, по одной в U добавляется одна вершина, по другой — минимум три вершины.

Количество вершин в дереве: $O(1{,}47^k)$

Сложность алгоритма: $O(1{,}47^k\, n^2)$

Оптимальный (на 07.10.16) алгоритм: $O(1{,}28^k\, n^2)$

Редукция данных

Параметрическая редукция данных (data reduction, kernlization).

Идея:

1) Выполнить предварительную обработку входных данных (граф), выделив эффективно решаемые фрагменты.
2) Оставшийся трудно решаемый фрагмент (*ядро*, *kernel*) — решаем экспоненциальным алгоритмом.

Если размер ядра окажется полиномиально зависящим от k, и не зависящим от n — получим FPT-алгоритм.

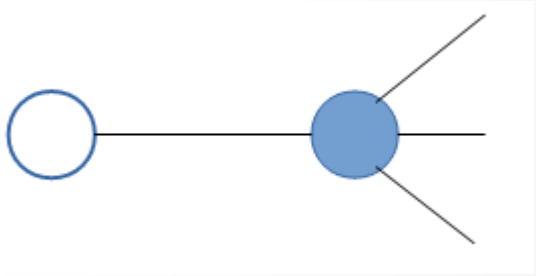
Редукция данных

Правило редукции № 1

Удалим все изолированные вершины.

Правило редукции № 2

Для каждой вершины степени=1 поместим в ВП её соседа, удалив из графа эту вершину и все инцидентные рёбра.

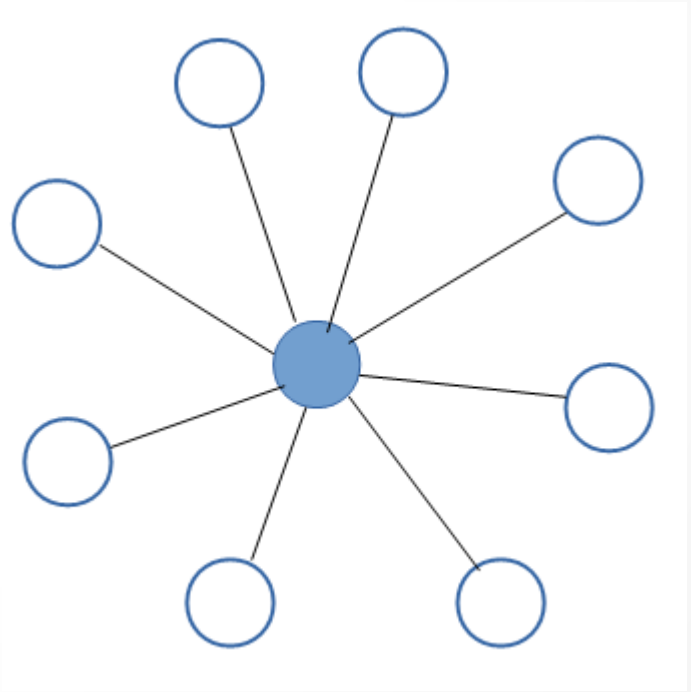


(Считаем, что надо найти *одно* минимальное покрытие).

Редукция данных

Правило редукции № 3

Если на графе есть вершина степени > k — поместим в ВП.



Редукция данных

Правила (1)-(3) применяем, пока есть такая возможность. В итоге получим граф G', в котором

- Степени всех вершин $\leq k$ (по правилу 3). Поэтому если на G' есть ВП размера $\leq k$, G' должен иметь не более k^2 рёбер.
- Степени всех вершин ≥ 2 (по правилам 1 и 2). Поэтому G' содержит не более k^2 вершин.

Таким образом, G' является *ядром* задачи.

Редукция данных

Определение

Пусть L — параметризованная задача с входными данными (x,k). *Параметрической редукцией данных* (kernelization) называется алгоритм, за полиномиальное время строящий для (x,k) новый экземпляр задачи (x',k'), такой что:

1. $k' \leq k$.
2. $|x'| \leq g(k)$, где g зависит только от k и не зависит от $|x|$
3. Решение для (x,k) существует $\Leftrightarrow$ существует решение для (x',k').

Редукция данных

Теорема

Для L существует параметрическая редукция данных $\Leftrightarrow$ L $\in$ FPT.

Доказательство

Часть '$\Rightarrow$' очевидна.

Докажем часть '$\Leftarrow$'. Пусть L $\in$ FPT. Тогда существует алгоритм A, решающий задачу (x,k) за время $O(f(k)\cdot n^c)$, $n=|x|$, c=const.

Построим алгоритм редукции A'.

Редукция данных

A' моделирует работу первых n^{c+1} шагов алгоритма *A*. Если за это время *A* найдёт решение, *A*' тоже возвращает это же решение. [Точнее, чтобы «соблюсти формальности», A' возвращает специально построенный по найденному решению экземпляр (*x*',*k*), либо возвращает «нет решения»]

Если *A* ещё не завершится, *A*' в качестве ядра возвращает исходные данные: (*x*,*k*). Поскольку A не завершился за n^{c+1} шагов, справедлива оценка:

$$f(k)n^c > n^{c+1} .$$

Отсюда получаем:

$$|x| = n < f(k).$$

Т.е. входные данные (*x*,*k*) и образуют ядро. □

Редукция данных

Замечание

Методы можно комбинировать. Например, на практике хороших результатов позволяет достичь чередование редукции данных и древовидного перебора ограниченной глубины.