Санкт-Петербургский государственный университет

Кафедра системного программирования

Определение реберной k-связности: детерминированные алгоритмы

Демченко Артем Евгеньевич, группа 22.Б11-мм

Санкт-Петербург
2025

Введение, определения

- Будем рассматривать **неориентированные** графы $G = (V, E)$.
- Граф называется реберно k-связным, если удаление любых (k−1) ребер оставляет граф связным.
- **Реберной связностью графа** называется

$$\lambda(G) = \max\{\, k \mid G \text{ реберно } k\text{-связен} \,\}.$$

- Для тривиального графа считаем: $\lambda(K_1) = 0$.
- Тривиальный граф — это граф, состоящий из одной вершины и не имеющий рёбер.

Пример

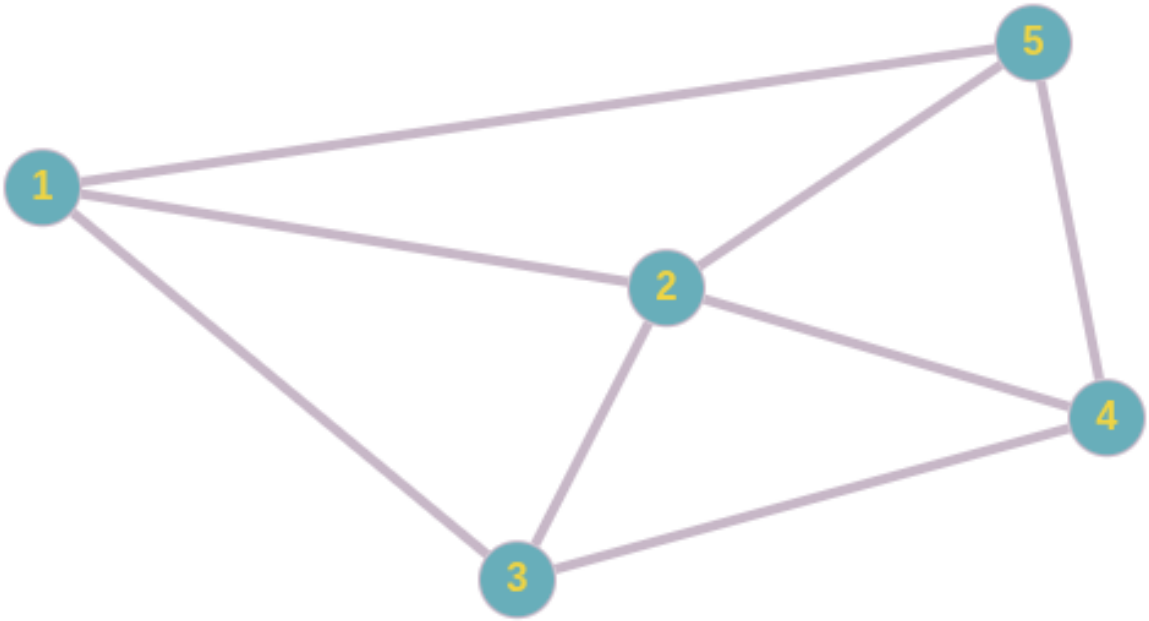

Рис.: Рёберная связность $\lambda = 3$

Применение алгоритмов определения k-связности

- Определение минимального числа каналов связи, при потере которых датацентр распадается на изолированные части.
- Проверка, сколько мостов в дорожной сети можно закрыть, прежде чем отдельные районы города потеряют соединение.
- Анализ резервных маршрутов в топологии облачного кластера, чтобы гарантировать доступность сервисов.
- Вычисление устойчивости сети электропередач: сколько линий можно вывести из строя без отключения потребителей.

Связь с теоремой Менгера

Theorem (Менгер, рёберная версия)

Между вершинами u и v существует L рёберно непересекающихся путей тогда и только тогда, когда после удаления любых $(L-1)$ рёбер в графе остаётся путь из u в v.

- Каждое ребро $\{u, v\} \in E$ заменяем на два ориентированных рёбра (u, v) и (v, u).
- Каждому ориентированному ребру приписываем пропускную способность $c(u, v) = 1$.
- Необходимо найти глобальный минимальный разрез в получившейся сети.

Решение задачи минимального разреза

После перехода к сети с единичными пропускными способностями задача сводится к нахождению **минимального разреза**. Существуют два основных подхода:

1. **Через max-flow:**
   - Для каждой пары вершин (s, t) вычисляем максимальный поток $\text{maxflow}(s, t)$.
   - По теореме max-flow/min-cut это значение равно минимальному s–t-разрезу, то есть $\lambda(s, t)$.
   - Глобальная рёберная связность: $\lambda(G) = \min_{s \neq t} \lambda(s, t)$.
2. **Алгоритм Штор-Вагнера:**
   - Итеративно объединяет вершины, вычисляя разрезы в упрощённой последовательности графов.
   - Находит **глобальный минимальный разрез** эффективнее, чем полный перебор всех пар.

Подход 1: через max-flow

- Для каждой пары вершин $(s, t) \in V$:
  - Вычисляем максимальный поток $\text{maxflow}(s, t)$ с помощью алгоритма Форда–Фалкерсона, Эдмондса–Карпа или push-relabel.
  - Значение потока совпадает с минимальным s–t-разрезом: $\lambda(s, t) = \text{maxflow}(s, t)$.
- Глобальная связность:

$$\lambda(G) = \min_{s \neq t} \lambda(s, t).$$

- Недостаток: требуется запускать max-flow для $O(|V|^2)$ пар, что дорого для больших графов.
- Преимущество: простая и концептуально наглядная связь с классическими алгоритмами потоков.

Определения (метод проталкивания предпотока)

- **Предпоток:** функция $f : V \times V \to \mathbb{R}$, для которой:
  1. $f(u, v) = -f(v, u)$ (антисимметричность)
  2. $f(u, v) \leq c(u, v)$ (ограничение пропускной способностью)
  3. $\forall u \in V \setminus \{s, t\} : \sum_{v \in V} f(v, u) \geq 0$ (ослабленное сохранение потока)
- **Избыточный поток:** $e(u) = \sum_{v \in V} f(v, u)$. Вершина u переполнена, если $e(u) > 0$.
- **Высота вершины:** функция $h : V \to \mathbb{Z}_+$:
  1. $h(s) = |V|$, $h(t) = 0$
  2. $h(u) \leq h(v) + 1$ для всех ребер (u, v) в остаточной сети.

Описание и идея алгоритма

- Дан граф $G = (V, E)$ с истоком s и стоком t, на рёбрах заданы пропускные способности $c(u, v)$.
- Поток $f(u, v)$ интерпретируем как «жидкость», текущую по трубам.
- Алгоритм работает с **предпотоком**, допускающим избыток в промежуточных вершинах.
- Интуитивная модель: резервуары на разных высотах, соединённые трубами.
- Пока есть переполненные вершины:
  - *Проталкивание (push)* — переливание избытка в более низкую вершину.
  - *Подъём (relabel)* — если проталкивать нельзя, вершина поднимается на 1.
- Когда переполненных вершин не остаётся, предпоток становится максимальным потоком.

Операция push

Условия применения проталкивания u в v:

- Вершина u переполнена: $e(u) > 0$.
- Остаточная пропускная способность $c_f(u, v) > 0$.
- Высота $h(u) = h(v) + 1$.

Действия:

- $\delta = \min(e(u), c_f(u, v))$ по ребру (u, v).
- Обновляем $f(u, v) := f(u, v) + \delta$, $f(v, u) := f(v, u) - \delta$.
- Уменьшаем $e(u)$ на δ, увеличиваем $e(v)$ на δ.

Операция relabel

Условия применения:

- Вершина u переполнена.
- Для всех соседей v с $c_f(u, v) > 0$ условие $h(u) \leq h(v)$.

Действие:

- Устанавливаем $h(u) := 1 + \min\{h(v) : c_f(u, v) > 0\}$.

Смысл: поднять «резервуар» так, чтобы появилось хотя бы одно допустимое ребро для проталкивания.

Схема алгоритма push–relabel

1. Инициализация: $h(s) = |V|$, $h(v) = 0$ для $v \neq s$, проталкиваем по рёбрам (s, v) максимальный поток.
2. Пока существует переполненная вершина $u \neq s, t$:
   - Если возможно, выполнить операцию push.
   - Иначе выполнить операцию relabel.
3. Когда переполненных вершин нет, полученный предпоток — максимальный поток.

Итог подхода 1

- Алгоритм работает за $O(V^2E)$ в общем случае.
- За счет выбора порядка обработки вершин можно добиться $O(V^3)$.
  - Используем стратегию **Relabel-to-Front**, которая задает определенный порядок обработки вершин.
- Получаем, что мы можем решить задачу определения k-связности за время $O(V^5)$.

Подход 2: алгоритм Штор-Вагнера

- Будем $n - 1$ раз повторять следующий процесс: находить минимальный разрез между какой-нибудь парой вершин s и t, а затем объединять эти две вершины в одну.
- Минимум по всем $n - 1$ найденным разрезам является ответом.

Алгоритм

- Старт: $A = \{a\}$ (любая вершина). Для $v \notin A$ поддерживаем $w_A(v) = \sum_{u \in A} w(u, v)$.
- Пока $A \neq V$: добавить в A вершину с максимальным $w_A(\cdot)$. Обновить $w_A(\cdot)$ для соседей.
- Пусть t — последняя добавленная вершина, s — предпоследняя.
- Обновить ответ весом разреза s-t; затем **склеить** s и t в одну вершину.

Асимптотика

1. Нахождение вершины с наибольшей w за $O(n)$, $n-1$ фаза по $n-1$ итерации в каждой. В итоге имеем $O(n^3)$.
2. Если использовать двоичные кучи, то асимптотика составит $O(nm \log n + n^2)$.

Корректность алгоритма

Theorem

Если добавлять в множество A по очереди все вершины, каждый раз добавляя вершину, наиболее сильно связанную с A, то пусть предпоследняя добавленная вершина — s, а последняя — t. Тогда минимальный s–t разрез состоит из единственной вершины — t.

Доказательство (шаг 1)

Рассмотрим произвольный s–t разрез C и покажем, что его вес не может быть меньше веса разреза, состоящего из единственной вершины t:

$$w(\{t\}) \leq w(C).$$

Пусть v — вершина, которую мы хотим добавить в A, тогда A_v — состояние множества A в этот момент. Пусть C_v — разрез множества $A_v \cup v$, индуцированный разрезом C. Назовем вершину v активной, если она и предыдущая добавленная вершина в A принадлежат разным частям разреза C, тогда для любой такой вершины, если мы докажем дополнительное условие ниже:

$$w(v, A_v) \leq w(C_v).$$

Доказательство (шаг 2)

То если t — активная вершина, для неё выполняется:

$$w(t, A_t) \leq w(C_t), \quad w(t, A_t) = w(\{t\}), \quad w(C_t) = w(C).$$

Получили утверждение теоремы. Для доказательства воспользуемся методом математической индукции. Для первой активной вершины v это неравенство выполнено, так как все вершины A_v принадлежат одной части разреза, а v — другой. Пусть неравенство выполнено для всех активных вершин до v, включая v, докажем его для следующей активной вершины u.

Доказательство (шаг 3)

$$w(u, A_u) = w(u, A_v) + w(u, A_u \setminus A_v). \quad (*)$$

Заметим, что

$$w(u, A_v) \leq w(v, A_v). \quad (**)$$

Вершина v имела большее значение w, чем u, так как была добавлена в A раньше. По предположению индукции:

$$w(v, A_v) \leq w(C_v).$$

Следовательно из $(**)$:

$$w(u, A_v) \leq w(C_v).$$

А из $(*)$:

$$w(u, A_u) \leq w(C_v) + w(u, A_u \setminus A_v).$$

Доказательство (шаг 4)

А из $(*)$ имеем:

$$w(u, A_u) \leq w(C_v) + w(u, A_u \setminus A_v).$$

Вершина u и $A_u \setminus A_v$ находятся в разных частях разреза C, значит $w(u, A_u \setminus A_v)$ равна сумме весов рёбер, которые не входят в C_v, но входят в C_u.
Таким образом:

$$w(u, A_u) \leq w(C_v) + w(u, A_u \setminus A_v) \leq w(C_u),$$

что и требовалось доказать.

Измерения, эксперименты

Были реализованы два подхода на языке Go 1.22, а именно:

- Max-flow (а именно push-relabel) для каждый пары вершин за $O(V^5)$.
- Алгоритм Штор-Вагнера без оптимизаций за $O(V^3)$.

Характеристики машины

- ОС Ubuntu 22.04.5 LTS
- Процессор 13th Gen Intel Core i7-13700H @ 2.4GHz
- L1d – 544KiB; L1i – 704KiB; L2 – 11.5MiB; L3 – 24MiB
- 32 GB RAM

Результаты с max-flow

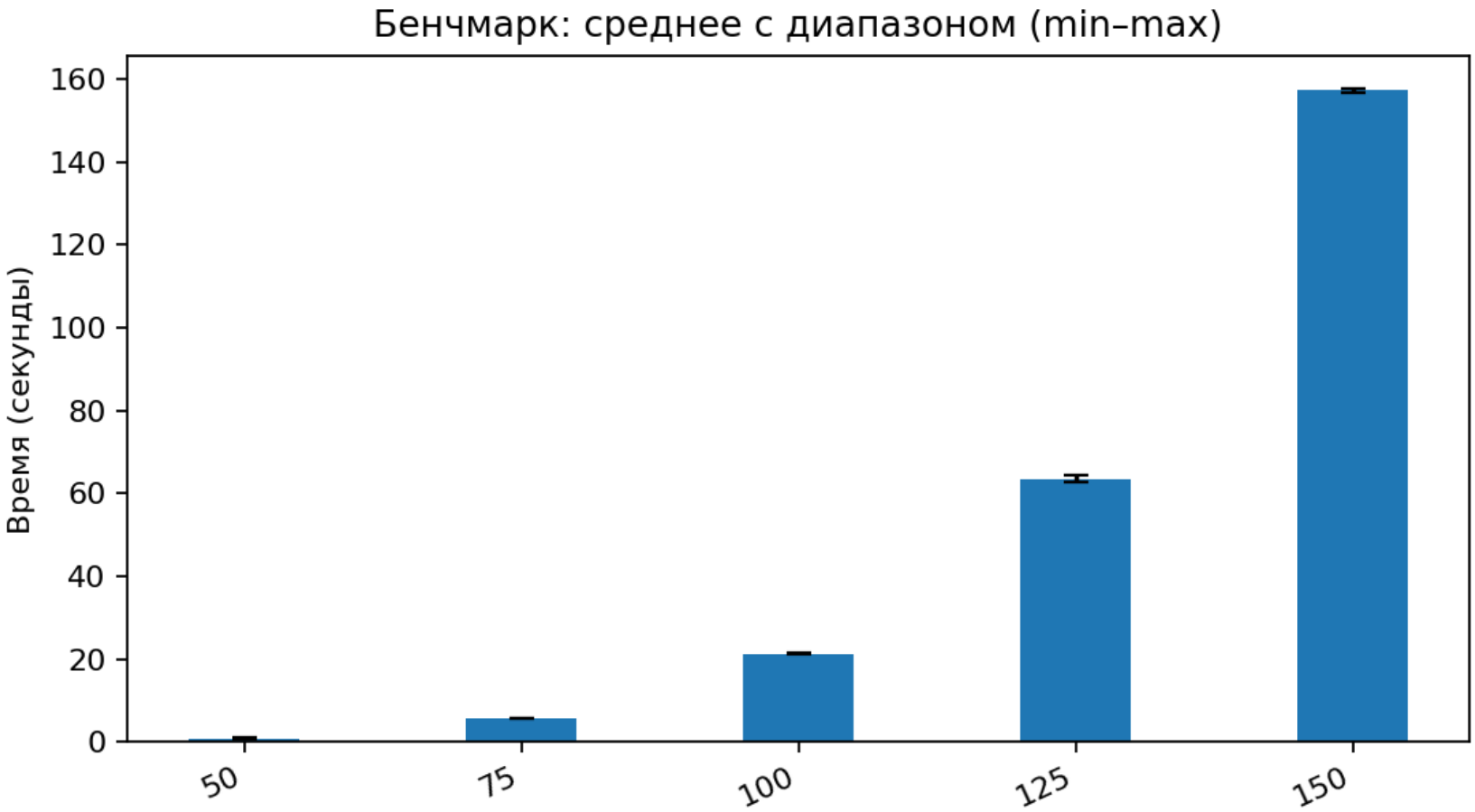

Рис.: Рёберная связность $\lambda = 3$

Результаты с алгоритмом Штор-Вагнера

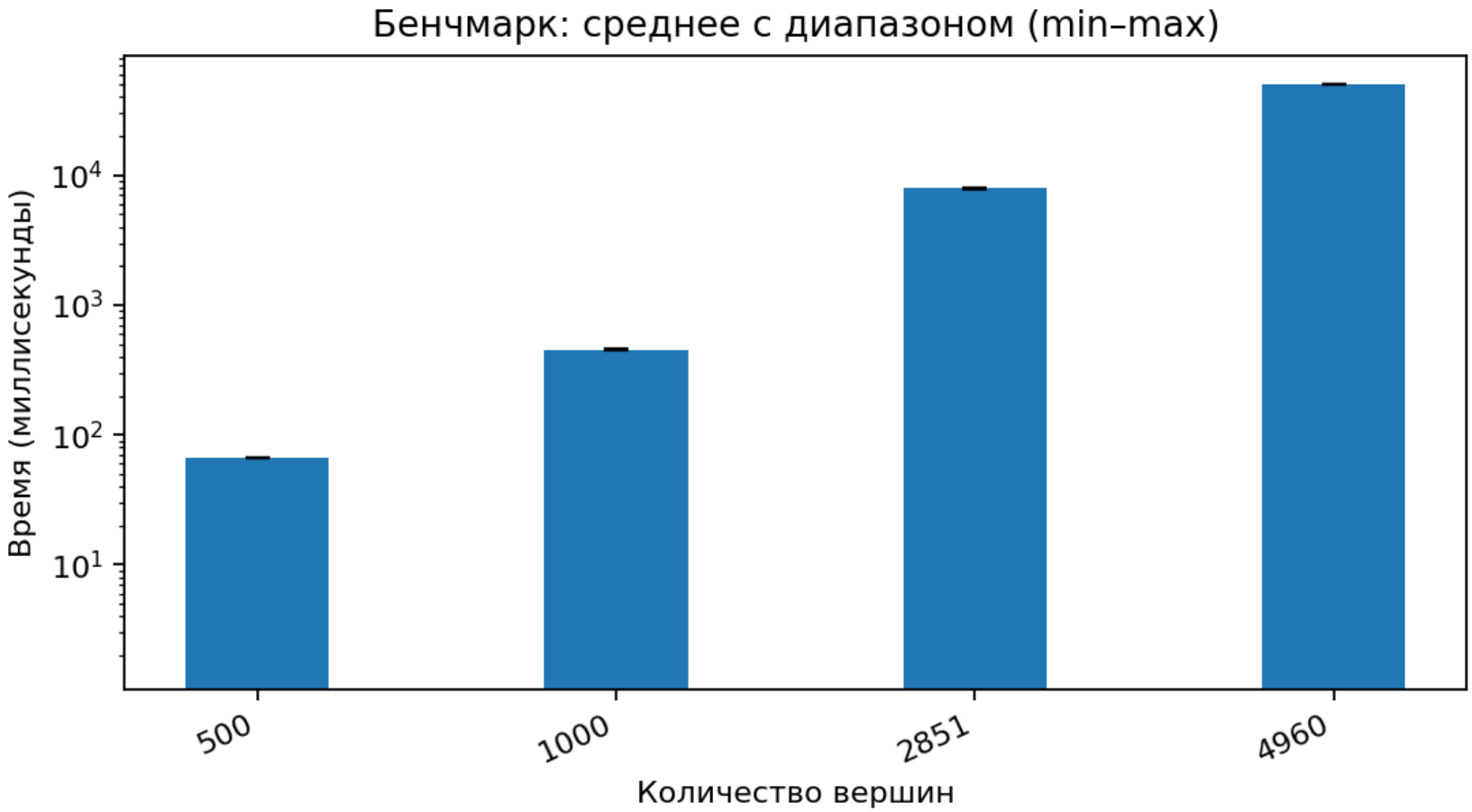

Рис.: Рёберная связность $\lambda = 3$