Network Flows

Потоки в сетях: максимальный поток, разрезы и MCMF

Конспект лекции: поток и остаточная сеть, теорема Форда — Фалкерсона, минимальный разрез, алгоритмы Эдмондса — Карпа и Диница, поток минимальной стоимости

---

1 Основные определения

Дан ориентированный граф, V — множество вершин, E — множество рёбер. Для каждого ребра e задана **пропускная способность** c_e. Выделены две вершины: **исток** s и **сток** t.

Определение 1. Потоком назовём набор величин f_e таких, что для всех $v \in V \setminus \{s, t\}$ выполнено условие сохранения

$$\sum_{u \in V} f_{uv} = \sum_{u \in V} f_{vu},$$

и для каждого ребра $0 \leqslant f_e \leqslant c_e$.

Величиной потока называется

$$|f| = \sum_{v \in V} f_{sv} = \sum_{v \in V} f_{vt}.$$

Мы хотим найти максимальный поток.

2 Остаточная сеть

Для каждого ребра $e \in E$ введём обратное ребро пропускной способности 0 и будем поддерживать соотношение $f_{uv} = -f_{vu}$. Такие рёбра позволяют нам *отменять* поток по некоторым рёбрам.

Определение 2. Остаточная сеть G_f — новый граф на том же множестве вершин, в котором пропускная способность ребра e равна $\hat{c}_e = c_e - f_e$.

Замечание. С учётом обратных рёбер условие корректности потока записывается как

$$\sum_{e \text{ — инц. } v} f_e = 0 \qquad \forall v \in V \setminus \{s, t\}.$$

сеть G остаточная сеть G_f

Рис. 1: Ребро с пропускной способностью 3, по которому пущено 2 единицы потока, порождает в остаточной сети прямое ребро остаточной пропускной способности 1 и обратное ребро пропускной способности 2.

3 Теорема Форда — Фалкерсона

Теорема 3 (Форд — Фалкерсон)**.** $|f| = |f_{\max}|$ тогда и только тогда, когда в G_f не существует пути $s \rightsquigarrow t$.

Доказательство. ($\Rightarrow$) Если путь есть, то поток вдоль этого пути можно увеличить хотя бы на 1, значит f не максимален.

($\Leftarrow$) Пусть $|f_{\max}| > |f|$. Рассмотрим $f_2 = f_{\max} - f$. Это корректный поток в G_f, причём $|f_2| > 0$, следовательно, в f_2 есть путь $s \rightsquigarrow t$. Этот же путь есть и в G_f — противоречие. □

Отсюда немедленно получается алгоритм:

```
while there is a path s ~> t in G_f:
   find such a path p
   push flow along p
```

Каждая итерация увеличивает поток хотя бы на единицу и стоит один обход графа, поэтому время работы

$$\boxed{O(|f_{\max}| \cdot (V + E)).}$$

4 Минимальный разрез

Разрезом графа называется пара $C = (V_1, V_2)$ такая, что $V_1 \cap V_2 = \emptyset$ и $V_1 \cup V_2 = V$. Величина разреза

$$|C| = \sum_{\substack{u \in V_1 \\ v \in V_2}} c_{uv}$$

есть сумма пропускных способностей рёбер, пересекающих разрез.

Замечание. Суммирование ведётся строго по $u \in V_1$, $v \in V_2$: *ориентация ребра важна*, обратные рёбра в величину разреза не входят.

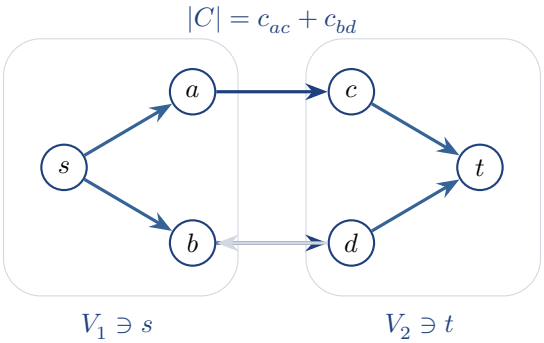

Рис. 2: Разрез $C = (V_1, V_2)$. В величину $|C|$ входят только рёбра, идущие из V_1 в V_2 (тёмные); ребро, идущее в обратную сторону (светлое), не учитывается.

Теорема 4 (Форд — Фалкерсон, расширенная формулировка)**.** Следующие утверждения эквивалентны:

1. $|f| = |f_{\max}|$;
2. в остаточной сети G_f нет пути $s \rightsquigarrow t$;
3. существует разрез $C = (V_1, V_2)$ такой, что $s \in V_1$, $t \in V_2$ и $|f| = |C|$.

Доказательство. (1) $\Leftrightarrow$ (2) — доказано выше.

(2) $\Rightarrow$ (3): возьмём в качестве V_1 все вершины, достижимые из s по ненасыщенным рёбрам, а в качестве V_2 — все остальные. Тогда $t \in V_2$, все рёбра из V_1 в V_2 насыщены, а по всем рёбрам из V_2 в V_1 поток нулевой, откуда $|f| = |C|$.

(3) $\Rightarrow$ (1): пусть $|C| = |f|$. Так как через любой разрез проходит весь поток, $|f_{\max}| \leqslant |C| = |f| \leqslant |f_{\max}|$, значит, $|f| = |f_{\max}|$. □

Как следствие получаем теорему о максимальном потоке и минимальном разрезе:

$$\boxed{|f_{\max}| = \min_C |C|.}$$

4.1 Алгоритм поиска минимального разреза

1. Находим максимальный поток f.
2. V_1 — все вершины, достижимые из s в остаточной сети G_f.
3. $V_2 = V \setminus V_1$.

4.2 Масштабирование

Во всех алгоритмах можно применять следующий приём. Перебираем 2^k от больших значений к меньшим и заменяем пропускные способности $\hat{c}_e$ на $\lfloor \hat{c}_e / 2^k \rfloor$. На каждом шаге мы имеем сеть с маленькими пропускными способностями, а значит, быстро делаем обновление; при переходе к следующему k пропускные способности удваиваются, и поток достраивается.

5 Алгоритм Эдмондса — Карпа

Будем вместо произвольного пути $s \rightsquigarrow t$ в G_f искать *кратчайший* (по числу рёбер, обычным обходом в ширину). Очевидно, алгоритм всё ещё корректен: это частный случай общей схемы из предыдущего раздела.

Теорема 5. В ходе алгоритма расстояние $d_f(s, x)$ от s до любой вершины x в G_f не убывает.

Доказательство. Пусть на какой-то итерации расстояние уменьшилось. Обозначим через f поток до итерации, через f' — после. Условие

$$d_f(s, x) > d_{f'}(s, x)$$

задаёт непустое множество вершин X. Рассмотрим $v \in X$ такую, что $d_{f'}(s, v) \to \min$, и предпоследнюю вершину u кратчайшего пути $s \rightsquigarrow v$ в $G_{f'}$. Тогда

$$d_{f'}(s, v) = d_{f'}(s, u) + 1 \geqslant d_f(s, u) + 1,$$

так как $u \notin X$ (иначе v не была бы минимальной) и, следовательно, $d_{f'}(s, u) \geqslant d_f(s, u)$. Разберём два случая.

1. Если $(u, v) \in G_f$, то $d_f(s, v) \leqslant d_f(s, u) + 1 \leqslant d_{f'}(s, v)$ — противоречие с $v \in X$.
2. Если $(u, v) \notin G_f$, то при переходе $f \to f'$ поток был увеличен по обратному ребру (v, u), то есть (v, u) лежало на кратчайшем пути в G_f. Значит,

   $$d_f(s, v) + 1 = d_f(s, u) \leqslant d_{f'}(s, u) = d_{f'}(s, v) - 1,$$

   откуда $d_f(s, v) \leqslant d_{f'}(s, v) - 2$ — снова противоречие.

□

Теорема 6. Алгоритм работает за $O(VE^2)$.

Доказательство. Назовём ребро e **критическим**, если $e \in p$, где p — выбранный путь $s \to t$, и $\hat{c}_e$ минимальна на пути p. На любом пути есть критическое ребро, а после увеличения потока все критические рёбра пути исчезают из G_f.

В следующий раз ребро $e = (u, v)$ может стать критическим только после того, как поток протолкнут по обратному к нему ребру. Пусть f — поток перед исчезновением (u, v), а f' — перед его появлением обратно. Тогда

$$d_f(s,u) + 1 = d_f(s,v) \leqslant d_{f'}(s,v) = d_{f'}(s,u) - 1 \quad \Longrightarrow \quad d_{f'}(s,u) \geqslant d_f(s,u) + 2.$$

Значит, ребро может стать критическим не более $|V|/2$ раз, откуда количество запусков BFS не более $|V| \cdot |E|$, а полное время работы составляет

$$\boxed{O(|V| \cdot |E|^2).}$$

□

6 Алгоритм Диница

Определение 7. Пусть есть сеть без циклов. **Блокирующим потоком** называется любой набор путей, который нельзя увеличить (при этом остаточная сеть не рассматривается).

6.1 Поиск блокирующего потока

1. Не более $|E|$ раз запускаем dfs: каждый успешный запуск насыщает хотя бы одно ребро.
2. В dfs можно поддерживать указатель на последнее ребро, которое стоит рассматривать. Если при рассмотрении ребра (u, v) и запуске dfs(v) ничего не нашлось — двигаем указатель.

```
bool dfs(v, t):
    if v == t: return true
    for (; it[v] < deg(v); ++it[v]): // it[v] -- the pointer
        (v, u) = edge[it[v]]
        if d[u] == d[v] + 1 and dfs(u, t):
            return true // do not move the pointer
    return false
```

Отсюда $O(VE)$: при рассмотрении ребра мы либо находим путь (длины не более V), либо двигаем указатель, а указатели суммарно проходят $O(E)$ шагов.

6.2 Собственно алгоритм

1. Считаем расстояния от s до остальных вершин в G_f.
2. Оставляем только рёбра, где $d_u + 1 = d_v$.
3. Ищем блокирующий поток в полученной сети.
4. Если существует путь $s \to t$ — повторяем.

Очевидно, алгоритм корректен по теореме Форда — Фалкерсона: он останавливается ровно тогда, когда в G_f не остаётся пути $s \rightsquigarrow t$. Более того, после каждой фазы расстояние от s до t строго возрастает, поэтому фаз не более $|V|$, и он работает за

$$\boxed{O(V^2 E).}$$

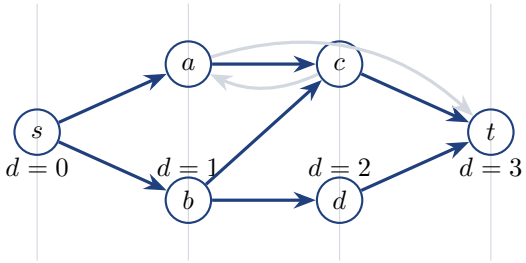

Рис. 3: Слоистая сеть: остаются только рёбра, ведущие с уровня d на уровень $d+1$ (тёмные). Рёбра, ведущие назад или через уровень (светлые), выбрасываются.

7 Min Cost Max Flow

Каждому ребру добавляем стоимость w_e. Хотим для фиксированного k найти поток размера k, минимизирующий $\sum_e w_e f_e$. Обратные рёбра получают вес $-w_e$.

7.1 Алгоритм

```
for i = 1..k:
    p = shortest_path(s, t) // Ford-Bellman: reverse edges have negative weight
    push 1 unit of flow along p
```

Ниже показано, что каждый промежуточный поток f_i имеет минимальную стоимость среди всех потоков величины i; в частности, ответ верен и для $i = k$. Всюду предполагается, что в исходной сети нет циклов отрицательного веса.

7.2 Сумма, разность и декомпозиция потоков

Для доказательства корректности нужны два технических инструмента. Всюду $\mathrm{val}(f) = |f|$ — величина потока, $\mathrm{cost}(f) = \sum_e w_e f_e$ — его стоимость, $w(P)$ и $w(C)$ — суммарные стоимости рёбер пути P и цикла C. Всё сказанное ниже применимо к любой сети со стоимостями, в том числе к остаточной: для неё роль c_e играют остаточные пропускные способности, а роль w_e — стоимости $\pm w_e$ прямых и обратных рёбер.

Определение 8. Пусть f — поток в G, а g — поток в остаточной сети G_f. Положим для каждого ребра e исходной сети

$$(f+g)_e = f_e + g_e - g_{\bar{e}},$$

где $\bar{e}$ — обратное к e ребро. Тогда $f+g$ — корректный поток в G, причём

$$\mathrm{val}(f+g) = \mathrm{val}(f) + \mathrm{val}(g), \qquad \mathrm{cost}(f+g) = \mathrm{cost}(f) + \mathrm{cost}(g)$$

(вторая формула верна именно потому, что обратному ребру приписан вес $-w_e$).

Обратно, пусть f и f' — два потока в G. **Разностью** $f' - f$ назовём набор величин на рёбрах G_f: на прямое ребро e кладём $\max(f'_e - f_e, 0)$, на обратное — $\max(f_e - f'_e, 0)$. Ограничения соблюдены, так как $f'_e - f_e \leqslant c_e - f_e = \hat{c}_e$ и $f_e - f'_e \leqslant f_e = \hat{c}_{\bar{e}}$. Это корректный поток *в остаточной сети* G_f, причём

$$\mathrm{val}(f'-f) = \mathrm{val}(f') - \mathrm{val}(f), \qquad \mathrm{cost}(f'-f) = \mathrm{cost}(f') - \mathrm{cost}(f), \qquad f + (f'-f) = f'.$$

Утверждение 9 (о декомпозиции)**.** Любой поток h (в исходной или в остаточной сети) раскладывается в набор путей $s \to t$ с положительными весами $\alpha_1, \dots, \alpha_m$ и циклов с положительными весами $\beta_1, \dots, \beta_r$, причём

$$\sum_i \alpha_i = \mathrm{val}(h), \qquad \mathrm{cost}(h) = \sum_i \alpha_i\, w(P_i) + \sum_j \beta_j\, w(C_j).$$

Доказательство. Пока есть ребро с положительным потоком, идём из него вперёд (и назад) по рёбрам с положительным потоком, пока не вернёмся в уже посещённую вершину или не упрёмся в s или t: по условию сохранения потока остановиться где-то ещё нельзя. В первом случае получаем цикл, во втором — путь $s \to t$. Вычитаем из h этот путь (цикл) с весом, равным минимуму потока на нём; хотя бы одно ребро обнуляется, поэтому шагов не более $|E|$. Обе формулы получаются суммированием по шагам. □

7.3 Корректность

Утверждение 10. Пусть N — сеть со стоимостями, в которой нет циклов отрицательной стоимости, P — путь из s в t минимальной стоимости $w(P)$ *в этой сети* N (кратчайший в смысле весов w, а не числа рёбер), и поток Δ получен проталкиванием γ единиц по каждому ребру P. Тогда в остаточной сети N_Δ тоже нет отрицательных циклов.

Доказательство. Заметим сразу, что $\mathrm{val}(\Delta) = \gamma$ и $\mathrm{cost}(\Delta) = \gamma\, w(P)$.

Пусть в N_Δ есть цикл C отрицательной стоимости, и пусть δ — минимальная остаточная пропускная способность на C. Пустив по каждому ребру цикла δ единиц, получим поток g в остаточной сети N_Δ, для которого

$$\mathrm{val}(g) = 0, \qquad \mathrm{cost}(g) = \delta\, w(C) < 0.$$

Рассмотрим поток $h = \Delta + g$ в самой сети N. По формулам сложения

$$\mathrm{val}(h) = \gamma, \qquad \mathrm{cost}(h) = \mathrm{cost}(\Delta) + \mathrm{cost}(g) < \gamma\, w(P).$$

С другой стороны, оценим $\mathrm{cost}(h)$ снизу через декомпозицию. Здесь существенно, что h — поток *в сети* N, а не в её остаточной сети: разложение использует только рёбра, на которых $h_e > 0$, то есть рёбра самой N, без обратных к ним. Поэтому каждый путь P_i декомпозиции — это путь $s \to t$ в N, то есть элемент того же множества, по которому брался минимум при выборе P; отсюда $w(P_i) \geqslant w(P)$. Точно так же каждый C_j — цикл в N, а отрицательных циклов в N по условию нет, поэтому $w(C_j) \geqslant 0$. Значит,

$$\mathrm{cost}(h) = \sum_i \alpha_i\, w(P_i) + \sum_j \beta_j\, w(C_j) \geqslant \Big(\sum_i \alpha_i\Big) w(P) = \gamma\, w(P),$$

что противоречит предыдущей строчке. □

Замечание (почему нельзя сказать «путь ровно один»). Соблазнительно прочитать $h = \Delta + g$ буквально: путь P плюс цикл C, значит, в разложении один путь и один цикл. Это верно ровно в одном случае — когда C целиком состоит из *прямых* рёбер сети N: тогда $h = \gamma \cdot P + \delta \cdot C$ — готовое разложение, и противоречие получается сразу, ведь $w(C) \geqslant 0$ по условию. Но C — цикл в *остаточной* сети N_Δ, а там есть обратные рёбра пути P (пропускной способности γ). Проходя по ним, цикл частично *отменяет* поток по P и уводит его в обход, и вид «путь плюс цикл» разрушается: например, если C проходит через s, то из s будут выходить два ребра с положительным потоком, и путей в разложении окажется не меньше двух. Поэтому число путей заранее неизвестно — и оценка выше его не использует: она усреднённая, $\sum_i \alpha_i\, w(P_i) \geqslant (\sum_i \alpha_i) w(P)$, и работает при любом их количестве.

Замечание (остаточная сеть остаточной сети). Если f — поток в G, а Δ — поток в G_f, то $(G_f)_\Delta = G_{f+\Delta}$: остаточная пропускная способность ребра e равна $(c_e - f_e) - \Delta_e + \Delta_{\bar{e}} = c_e - (f + \Delta)_e$, а стоимости рёбер на всех уровнях одни и те же (w_e у прямого, $-w_e$ у обратного).

Следствие. Пусть в G нет отрицательных циклов. Тогда ни на одном шаге алгоритма отрицательных циклов не появится. Индукция по номеру итерации: база — $G_{f_0} = G$;

переход — алгоритм берёт кратчайший путь P именно в $N = G_{f_i}$, где по предположению индукции отрицательных циклов нет, и проталкивает по нему поток Δ, так что по утверждению выше их нет и в $N_\Delta = G_{f_{i+1}}$.

Замечание. Именно поэтому утверждение сформулировано для произвольной сети N: применяется оно не к G, а к *текущей остаточной сети*. Внутри доказательства P и все пути P_i декомпозиции живут в одной и той же сети N — сравнивать их законно. Если бы P выбирался в G_{f_i}, а декомпозиция проводилась в G, вывод $w(P_i) \geqslant w(P)$ был бы неверен: в остаточной сети есть обратные рёбра стоимости $-w_e$, и путь в ней может быть дешевле любого пути в G. Отметим также, что для *произвольного* потока f утверждение «в G_f нет отрицательных циклов» неверно и всегда требует обоснования — здесь оно получено индукцией, опирающейся на то, что каждый шаг идёт по кратчайшему пути.

Утверждение 11. Пусть f — поток. В G_f нет отрицательных циклов тогда и только тогда, когда f имеет минимальную стоимость среди всех потоков величины $\mathrm{val}(f)$.

Доказательство. ($\Leftarrow$) Пусть f минимален по стоимости среди потоков величины $\mathrm{val}(f)$. Если бы в G_f был отрицательный цикл C, мы бы определили поток g в G_f, пустив по каждому его ребру $\delta > 0$ единиц, где δ — минимальная остаточная пропускная способность на C. Тогда $\mathrm{val}(g) = 0$, но $\mathrm{cost}(g) < 0$, то есть $\mathrm{val}(f+g) = \mathrm{val}(f)$ и $\mathrm{cost}(f+g) < \mathrm{cost}(f)$ — противоречие с минимальностью.

($\Rightarrow$) Докажем контрапозицию. Пусть f не минимален: существует поток f' с $\mathrm{val}(f') = \mathrm{val}(f)$ и $\mathrm{cost}(f') < \mathrm{cost}(f)$. Рассмотрим их разность $g = f' - f$ — это поток *в остаточной сети* G_f, причём

$$\mathrm{val}(g) = \mathrm{val}(f') - \mathrm{val}(f) = 0, \qquad \mathrm{cost}(g) = \mathrm{cost}(f') - \mathrm{cost}(f) < 0.$$

Разложим g на пути и циклы. Сумма весов путей равна $\mathrm{val}(g) = 0$, а веса положительны, значит, путей в разложении нет и $\mathrm{cost}(g) = \sum_j \beta_j w(C_j)$. Эта сумма отрицательна, поэтому хотя бы для одного цикла $w(C_j) < 0$ — это и есть отрицательный цикл в G_f. □

Вместе эти два утверждения показывают, что на всём протяжении работы алгоритма поток f остаётся минимальным по стоимости для своей текущей величины. Это и доказывает корректность: если поток минимален по стоимости на каждом шаге, то он минимален и в конце. Заметим также, что алгоритм — частный случай общей схемы поиска максимального потока, поэтому при $k = |f_{\max}|$ полученный поток максимален.

7.4 Потенциалы Джонсона

Запускать алгоритм Форда — Беллмана на каждой из k итераций дорого. Хочется пользоваться Дейкстрой, но обратные рёбра имеют отрицательный вес. Заведём потенциалы p_v и потребуем

$$p_u + w_{uv} - p_v \geqslant 0 \quad \Longleftrightarrow \quad p_v \leqslant p_u + w_{uv},$$

то есть p ищем как кратчайшие расстояния от фиктивной вершины, соединённой рёбрами нулевого веса со всеми остальными (один запуск Форда — Беллмана). Далее на графе с новыми весами

$$\boxed{\widehat{w}_{uv} = p_u + w_{uv} - p_v \geqslant 0}$$

запускаем Дейкстру. Кратчайшие пути в новых весах совпадают с кратчайшими путями в старых, так как вес пути $u \rightsquigarrow v$ меняется на константу $p_u - p_v$. После проталкивания потока обновляем потенциалы новыми кратчайшими расстояниями.