Лекция 5

Максимальное паросочетание

(Конспект: О. Медведев, В. Трофимов)

Disclaimer: *Конспект приводится “as is”, я его не смотрел. – – Э.А.*

5.1 Поиск максимального по мощности паросочетания в произвольном графе

В следующем разделе рассматривается более общая задача. Поэтому этот раздел можно было бы опустить, но интуиция, полученная здесь на более простой задаче, может быть полезна при прочтении следующего раздела.

Рассмотрим граф без кратных ребер и петель. Под паросочетанием будем понимать его подграф, в котором каждая вершина имеет степень 1 (т.е. все вершины разбиты на пары, соединенные ребром). Под максимальным - максимальное по количеству ребер (равно как и вершин).

Описываемый далее алгоритм основан на понятии дополняющего пути. Пусть а нашем графе фиксировано паросочетание P, тогда.

def 1. Дополняющий путь - это простой путь нечетной длинны, в котором ребра, принадлежашие P чередуются с ребрами, не пренадлежащими P, первое ребро, первая и последняя вершины не пренадлежат P. (Например, путь вдоль стрелочек на рис. 5.1 слева).

Ясно, что если мы найдем дополняющий путь A, то паросочетание $P' = P \oplus A$ (симметрическая разность множеств ребер P и A) будет содержать на одно ребро больше (и тоже будет паросочетанием).

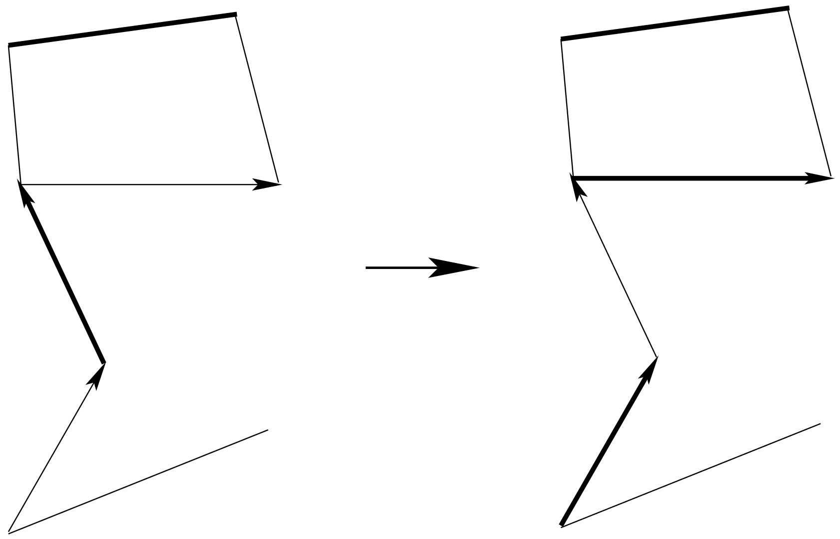

Рис. 5.1: Увеличение паросочетания засчет "инвертирования"дополняющего пути. На этом и всех последующих рисунках жирные ребра - из паросочетания, а худые - нет.

Lemma 1. В графе есть дополняющий путь $\Leftrightarrow$ текущее паросочетание P не максимально.

Доказательство.

$\Rightarrow$: выше пояснялось, как при помощи дополняющего пути увеличить P

$\Leftarrow$: Пусть дополняющего пути нет, P не максимально, а P' - максимально. Рассмотрим $M = P \oplus P'$ - подграф из всех ребер, принадлежащих только P или P' вместе с инцидентными им вершинами. Степень любой вершины в нем ≤ 2 (т.к. и P, и P' - паросочетания) - значит это объединение нескольких циклов и путей. Более того, и в циклах, и в путях ребра из P и P' чередуются. Значит, во всех циклах ребер P и P' поровну, но всего ребер в P' больше, а при взятии симметрической разности из P и P' выкинуто одинаковое количество, значит есть путь, где ребер из P' больше - это и есть дополняющий путь (понятно, что из первой вершины пути в исходном графе не выходило других ребер из P', т.к. P' - паросочетание, и никаких из P, т.к. она оказалась первой).

Теперь понятно, как строить паросочетание: взять пустое, и расширять его засчет дополняющих путей, пока они есть. То что получится - ответ.

Как найти дополняющий путь? Мы модифицируем для этой цели какой-нибудь поиск (например, в ширину). Он будет искать не какой-нибудь путь, а полосатый (т.е., такой, у которого первое ребро не из P, следующее - из P, и т.д.), т.к. только у такого есть шансы оказаться дополняющим. Получим алгоритм, который находит путь иногда, а потом улучшим так, чтобы путь находился всегда, когда он есть.

Итак, пусть V - вершины графа, E - ребра. V_P - вершины текущего паросочетания P, E_P - ребра. Алгоритм делает так:

- Среди еще необработанных вершин берем $\forall a : \neg a \in V_P$ и метим ее как `even` (раз мы ищем не все пути, а только полосатые, то даже связный граф не обязан уложиться в одно дерево поиска путей, так что у нас вырастет целый лес). Кладем a в очередь.

- Пусть u - сдедующая из очереди, v - ее сосед.
  `If` v не обработана `then begin`
  `If` $\neg v \in V_P$ `then`
  Путь от a до v - дополняющий
  `else`
  Пометим v как `odd`, ее жениха $v' \in V_P$ как `even` и отправим его в очередь - таким образом все вершины в очереди будут `even`, а любой путь от a до листа дерева окажется полосатым.
  `end else begin`
  `If` v - `odd then`
  единственное возможное продолжение полосатого пути из v можно считать уже рассмотренным - ничего не делаем.
  `else`
  Ясно, что эта `even` вершина - из нашего дерева (иначе у нас просто есть дополняющий путь от a до корня дерева, в котором лежит v и он был бы найден раньше при выращивании того дерева. Значит, мы нашли “почти” полосатый цикл нечетной длины(такой, как цикл на рис. 5.2). Назовем цикл такого типа “цветочек”. В принципе, он нам не нужен, так что ничего не делаем.
  `end`

- Идем в пункт 1.

На рис. 5.3 изображен пример обхода дерева таким способом. Имеется очевидный путь 145326, который при таком обходе не найден. На данном рисунке можно удалить ребро 12, после чего путь будет найден. Заметим, однако, что при этом также пропадет цветочек 123541. Давайте обоснуем, что все зло именно в цветочках. Для этого модифицируем самый последний `else` нашего алгоритма - найденный там цветочек мы будем стягивать в точку, метить ее как `even` и продолжать выполнение алгоритма уже в факторграфе. Заметим, что из цветочка выходит не более

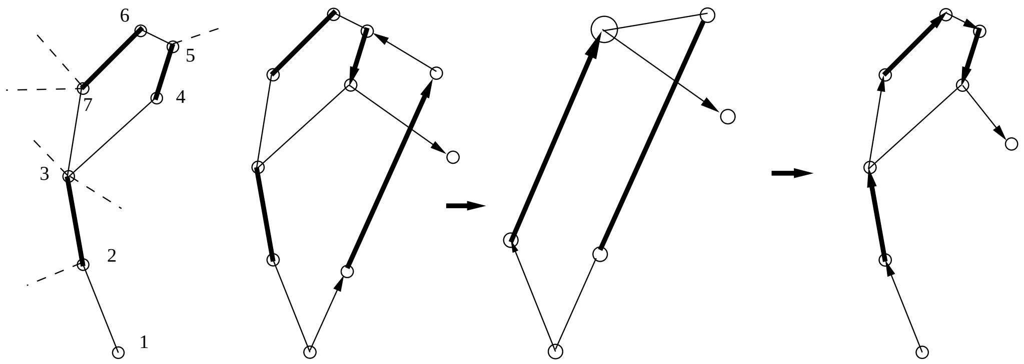

Рис. 5.2: Общий вид цветочка со стебельком (слева). Что случится с дополняющим путем, проходящим через цветочек при переходе к факторграфу и обратно (справа). Заметим, что при факторизации наш путь пришлось несколько извратить, иначе бы он перестал быть дополняющим.

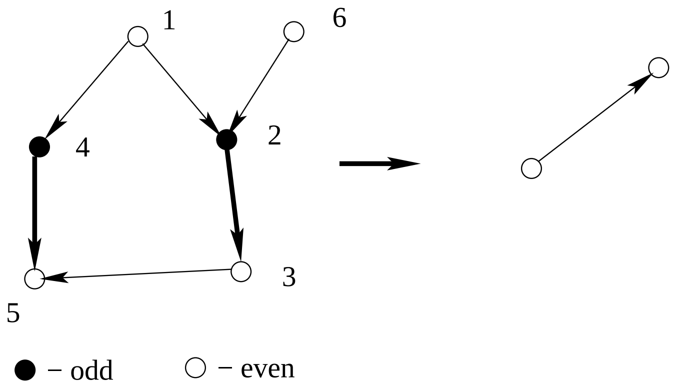

Рис. 5.3: Цветочек и факторцветочек

одного ребра P (все `even` вершины, кроме одной содержатся в цветочке вместе с парными `odd`, а значит, и ребром из P), поэтому ничто не мешает нам объявить его новой `even` вершиной. После такого действия из нее будут выходить необработанные ребра, точнее, необработанными окажутся только некоторые ребра бывшей вершины u (поскольку все остальные `even` вершины цветочка уже были обработаны нашим поиском в ширину) и все ребра всех `odd` вершин из цветочка, не пренадлежавшие паросочетанию. Алгоритм просто продолжает их обрабатывать.

Lemma 2. Дополняющий путь есть в факторграфе $\Leftrightarrow$ есть в исходном графе.

Очевидно, если в исходном графе дополняющий путь был, то он заканчивался вне цветочка, а значит, последнее из ребер, по которому этот путь из цветочка выходил можно объявить первым ребром пути в новом графе. Возможно, придется поменять начало пути, с тем чтобы не оказалось, что путь входит в цветочек и выходит из него не по ребру из P. Это всегда можно сделать, пропустив начало пути из корня по “стеблю”, на котором цветочек растет, а затем - по цветочку в нужную сторону.

Для пути в факторграфе можно рассмотреть вершину (в исходном графе), через которую он может войти в цветочек и ту, через которую он может выйти, после чего удастся пропустить отрезок пути между ними в какую-то сторону по цветочку, не нарушив свойств пути.

Lemma 3. Если дополняющего пути не нашлось, то в самом последнем факторграфе (т.е, последнем, с которым работал алгоритм) нет смежных `even` вершин.

Очевидно: иначе наш алгоритм нашел бы еще один цветочек и стянул бы его. Заметим, что даже если после стягивания получится новое ребро между `even` вершинами, то оно (в исходном графе) проходило между `odd` и `even` вершинами и мы его сразу найдем, обрабатывая все такие ребра.

Lemma 4. Если в исходном графе был путь, то наш алгоритм его нашел.

Пусть не нашел. Значит все вершины разбиты на `even` и `odd`, и их больше одной (дополняющего пути внутри цветочка в принципе не бывает, т.к. не достаточно вершин не из P). Заметим, что первая и последняя вершины пути - `even`, т.к. они не принадлежат паросочетанию. Рассмотрим граф после всех факторизаций и придем в противорчие с предыдущей леммой, поскольку вершин в пути четное количество, однако, `even` и `odd` вершины обязаны чередоваться.

Из последней леммы следует корректность нашего алгоритма.

Замечание. Автор этого конспекта обнаружил нескольких людей, искренне убежденных в том, что задача поиска паросочетания в произвольном графе NP-полна. В попытках выяснить причины такого заблуждения он узнал, что NP-полной является задача о поиске минимального по мощности вершинного покрытия в произвольном графе (вершинным покрытием называется такое $V' \subset V$, что $\forall e \in E$ один из концов e принадлежит V' - e "покрыто" каким-то элементом V'). В случае же двудольного графа ("классическая" задача о паросочетании), где вершинное покрытие еще называется контролирующим множеством, эти две задачи являются двойственными и обе быстро решаются.

5.2 Поиск максимального по весу паросочетания в произвольном графе

Задачу о поиске максимального паросочетания можно записать в виде следующей задачи линейного программирования. Заведем по переменной x_e для каждого ребра из нашего графа. Будем считать, что $x_e = 1$ если ребро принадлежит паросочетанию, и $x_e = 0$ в противном случае. Тогда для этой задачи можно написать следующие граничные условия: $\forall e \in E\ x_e \geq 0$; $\forall v \in V\ \sum_{e \in v} x_e \leq 1$.

Целевой функцией в этой задаче будет являться $\sum_{e \in E} W(e) x_e$. Мы будем искать ее максимум при данных граничных условиях. Решение данной задачи не обязательно даст нам максимальное паросочетание, так как x_e не обязательно является нулем или единицей, а 0-1 программирование является NP-трудным. Но если мы построим паросочетание и докажем, что на нем реализуется максимум целевой функции, то мы решим исходную задачу. Добавим к нашим граничным условиям условия вида $\forall r\ \forall s \subset V\ (|s| = 2r+1 \implies \sum_{e \in s} x_e \leq r)$. Если мы найдем паросочетание, максимизирующее целевую функцию для изначальной задачи линейного программирования, то оно будет являться решением и для дополненной задачи, так как для паросочетания новое условие, очевидно, выполняется. Напишем теперь двойственную задачу линейного программирования к только что полученной.

$$
\begin{array}{ll}
\max < c, x > & \min < b, y > \\
\quad x_i \geq 0; & \quad Ax \leq b; \\
\quad A^T y \geq c; & \quad y_j \geq 0.
\end{array}
$$

Если вместе с паросочетанием мы найдем и решение двойственной задачи, такое, что $< c, x_* >=< b, y_* >$, то мы решим исходную задачу. Переменные в двойственной задаче будут двух видов: отвечающие условиям на вершины y_v и отвечающие условиям на подмножества вершин z_s. Напишем граничные условия для двойственной задачи: $y_v \geq 0; z_s \geq 0$

$$\forall e = (v_1, v_2) \quad y_{v_1} + y_{v_2} + \sum_{e \in s} z_s \geq W(e) \qquad (*)$$

При этих условиях нам необходимо минимизировать $\sum y_v + \sum z_s|s|/2$. Напишем для нашей пары двойственных задач условия дополнительности: $y_v = 0$ для $v \notin M$ (M — паросочетание); $z_s = 0$ для всех s, для которых неравенство из прямой задачи ЛП выполняется, как строгое; (*) обращается в равенство для $\forall e \in M$.

Допустим, что мы нашли паросочетание и решение двойственной задачи такие, что выполнены условия дополнительности, тогда значения целевых функций совпадут. В самом деле, если сложить все неравенства вида (*) то с учетом условий дополнительности получим $\sum_{v \in V} y_v + \sum_s z_s|s|/2 = \sum_{e \in M} W(e) =< x, w >$.

Теперь мы будем строить последовательность графов $G_0, G_1, \ldots, G_m$ и последовательность паросочетаний $M_0, M_1, \ldots, M_m$ такие, что $G_0 = G$ — исходному графу, $M_m = M$ — искомому паросочетанию, G_i получается из $G_i - 1$ при помощи операции стягивания цветочка — стягивания в вершину цикла нечетной длины, у которого максимальное возможное число ребер принадлежат паросочетанию. В наших графах будут рассматриваться не только веса на ребрах, но и веса на вершинах. В ходе наших преобразований мы будем следить за выполнением следующих свойств.
(a) $G = G_0$.
(b) $W(v) \geq 0; \forall e = (v_1, v_2) : W(v_1) + W(v_2) \geq W(e)$.
(c) B_i — цветок в G_i.
(d) Для всех ребер цветочка $W(v_1) + W(v_2) = W(e)$ (ребра точные).
(e) q^i (q из графа G_i) $\in B_i$ - ребро с наименьшим весом в цветочке. q^i не из M_i (если есть).
(f) B_i стягивается в $u^i + 1$.
(h) $W(u^i + 1) \leq W(q^i)$.
(i) $W(e^i+1) = W(e^i)+W(q^i)-W(v^i)$, где e^i соединяет $v^i \in B_i$ с вершиной не из цветочка.
(j) В последнем паросочетании все ребра точные.
(k) $\forall v^m \notin M_m : W(v^m) = 0$.

Покажем, что соблюдение условий $a, \ldots, k$ даст нам решение прямой и двойственной задачи. При этом x_i - полученное паросочетание, а z_s, y_v

еще надо определить. $d_i = W(q^i) - W(u^i + 1)$ для стягиваемых цветочков, и ноль иначе. $Z_{B_i} = 2d_i$ и ноль иначе, $y_v = W(v) - \sum_{v \in s} d_s$.

Условие $z_s = 0$ для ненасыщенных подмножеств выполняется, так как цветочки всегда насыщены, а только на них $Z_s > 0$.

$\forall e$ из исходного графа где-то закончилось $e \to e^j$. Вес ребра изменяется, если один его конец лежит в стягиваемом цветочке, а другой не лежит. Если рассмотреть независимо стягивания каждого из концов ребра, то при условии равенства в (h) получим $W(e^j) = W(e) + W(v_1^j) + W(v_2^j) - W(v_1) - W(v_2) + \sum dB_k$, где последняя сумма идет по всем цветочкам, содержащим ровно один из концов ребра. $W(e^j) - W(v_1^j) - W(v_2^j) \leq 0$, и равно нулю, если ребро попало в паросочетание. Значит мы получаем $W(e) \leq W(v_1) + W(v_2)) - \sum dB_k$. В случае попадания ребра в ответ последнее неравенство обращается в равенство. Отсюда следует третье условие дополнительности. Все z_s неотрицательны в силу (h). Все y_v неотрицательны, поскольку $y_v = W(v) - \sum$, а это не меньше веса последней вершины, в которую стянется v. Упражнение. $y_v = 0$ для $v \notin M$.

В процессе описания алгоритма мы будем следить за двумя полуинвариантами. Во-первых, количество вершин, не удовлетворяющих условию завершения (k) будет не увеличиваться с каждым шагом алгоритма и, вовторых, если первый полуинвариант не изменяется, то увеличивается количество ребер, добавленных к очередному лесу поиска.

В начале алгоритма последовательность графов состоит только из $G_0 = G$, начальное паросочетание пусто и веса вершин достаточно большие, чтобы выполнялось условие (b). Алгоритм берет произвольную вершину последнего графа для которой не выполняется условие (k). Начнем, как и в случае невзвешенного графа, строить из нее лес поиска дополняющих цепей. Использоваться при этом будут только точные ребра. Вершины, в которые мы попадем по ребру из паросочетания, и корни деревьев мы назовем четными, а остальные — нечетными. В ходе построения дерева обязательно возникнет одна из следующих ситуаций:

1. Мы пришли в еще одну невзятую вершину. В этом случае мы инвертируем вдоль пути из корня в эту вершину принадлежность всех ребер паросочетанию. После этого наш первый полуинвариант уменьшится, и можно начинать новый шаг алгоритма. На самом деле нетрудно понять, что при этой замене увеличится и суммарный вес паросочетания, так как все ребра на пути точные. Поскольку мы не меняет никаких весов, то никакие свойства, за которыми мы следим, не нарушатся.
2. Мы пришли во взятую вершину с весом ноль. В этом случае делаем все так же, как и в предыдущем пункте, с той лишь разницей, что количество ребер в паросочетании не изменяется.
3. Мы нашли цветочек, то есть точное ребро из четной вершины в дру-

гую четную вершину. В этом случае мы заводим новый граф, в котором этот цветочек будет стянут в вершину $u^i + 1$ с весом $W(q^i)$, и переносим на этот новый граф наш лес поиска. Цветок в этом лесе стянется в четную вершину, и все ребра в новом дереве останутся точными, так как вес ребер из окрестности цветочка изменяется по свойству (i), а ребра не из окрестности цветочка мы не трогаем. При этом шаге второй полуинвариант увеличивается.
4. Мы не можем больше расширять наш лес поиска. В этом случае мы меняем веса у вершин. Мы постепенно увеличиваем веса всех нечетных вершин и с той же скоростью уменьшаем веса всех четных вершин, пока не возникнет одна из следующих ситуаций.
a) Возникло новой точное ребро из четной вершины. Оно не может идти в нечетную вершину, так как сумма весов четной и нечетной вершин не меняется. Значит его можно добавить к нашему лесу поиска и увеличить второй полуинвариант.
b) Вес на вершине стал равным нулю. Если это корень, то первый полуинвариант уменьшился, иначе возникает ситуация из пункта 2.
c) Перестало выполняться неравенство (h) $W(u^i + 1) \leq W(q^i)$. Это могло случиться только в нечетной вершине. В этом случае мы раскрываем обратно наш цветочек и откатываемся к предыдущему графу. Поскольку этот цветок мог быть не последним стянутым, то формально нам надо растянуть его во всей последовательности графов, начиная с его стягивания. Так как мы стягиваем цветочки в четные вершины, а раскрываем из нечетных, то мы не зациклимся.

Поскольку половина ребер в лесе поиска из паросочетания, алгоритм совершит не более V^2 шагов. После того, как мы нашли паросочетание в последнем графе, нам остается "пронести" его в изначальный граф, раскрывая по пути все цветочки. При раскрытии цветочка появляется не более одной вершины, принадлежащей паросочетанию (стебель), значит между остальными его вершинами есть путь из $2k - 1$ ребер, из которых k берутся в паросочетание.